

Introdução à acessibilidade urbana

um guia prático em R

Rafael H. M. Pereira, Daniel Herszenhut

2022-09-30T00:00:00+00:00

Table of contents

Apresentação	5
Organização do livro	5
Licença de uso	6
Como citar o livro	6
Agradecimentos	6
 I SEÇÃO 1: Introdução à acessibilidade urbana	 7
1 O que é acessibilidade?	9
1.1 Definição de acessibilidade urbana	9
1.2 Por que acessibilidade importa?	9
1.3 Diferença entre micro e macro acessibilidade	10
1.4 Diferença entre acessibilidade e mobilidade urbana	11
 2 Indicadores de acessibilidade	 12
2.1 Medidas baseadas em lugares	12
2.1.1 Mínimo custo de viagem	12
2.1.2 Medida cumulativa de oportunidades	13
2.1.3 Medidas gravitacionais	14
2.1.4 Indicadores de acessibilidade com competição: <i>floating catchment area</i> .	15
2.2 Medidas baseadas em pessoas	15
 II SEÇÃO 2: Calculando acessibilidade	 17
3 Calculando acessibilidade urbana em R	19
3.1 Cálculo da matriz de tempo de viagem	19
3.1.1 Instalação do r5r	19
3.1.2 Dados necessários	20
3.1.3 Calculando a matriz de tempo de viagens	21
3.2 Cálculo de acessibilidade	24
3.3 Cálculo de acessibilidade com o r5r	27
3.4 Análises de acessibilidade	29

III	SEÇÃO 3: Dados de transporte público	34
4	Dados GTFS	36
4.1	Estrutura dos arquivos de GTFS	36
4.1.1	agency.txt	38
4.1.2	stops.txt	39
4.1.3	routes.txt	40
4.1.4	trips.txt	42
4.1.5	calendar.txt	43
4.1.6	shapes.txt	43
4.1.7	stop_times.txt	44
4.1.8	frequencies.txt	45
4.2	Onde encontrar GTFS de cidades brasileiras	47
5	Manipulação e visualização de dados GTFS	48
5.1	Leitura e manipulação básica de arquivos GTFS	48
5.2	Cálculo de velocidade das linhas	51
5.3	Combinação e filtragem de <i>feeds</i>	52
5.4	Mapeamento do <i>headway</i> das linhas	65
IV	SEÇÃO 4: Avaliação de impacto	70
6	Comparando a acessibilidade entre dois cenários de transporte	72
6.1	Modificação do arquivo GTFS	72
6.2	Cálculo das matrizes de tempo de viagem	73
6.3	Cálculo da acessibilidade nos cenários antes e depois	77
6.4	Visualização dos níveis e da diferença de acessibilidade	77
6.5	Distribuição da acessibilidade entre grupos sociais	77
V	SEÇÃO: 5: Dados do Projeto AOP	78
VI	Quais dados estão disponíveis?	80
	Escopo dos dados:	81
	Dados de população, empregos e serviços públicos:	81
	Dados de acessibilidade urbana:	81
7	Dados de população e socioeconômicos	82
7.1	Download dos dados	82
7.2	Mapa de população total	83
7.3	Mapa de população por cor	84
7.4	Mapa de população por renda	85

8	Dados de distribuição espacial de oportunidades	87
8.1	Download dos dados	87
8.2	Mapa de empregos	89
8.3	Mapa de escolas	89
8.4	Mapa de estabelecimentos de saúde	90
8.5	Mapa de CRAS	91
9	Estimativas e mapas de acessibilidade	93
9.1	Download dos dados	93
9.2	Mapa do tempo para acessar o hospital mais próximo	99
9.3	Mapa da quantidade de oportunidades acessíveis	100
9.4	Desigualdades de acesso a oportunidades	102
	Referências bibliográficas	107

Apresentação

O objetivo deste livro é apresentar uma introdução sobre os conceitos e habilidades práticas necessárias para fazer estudos e avaliações de impacto sobre acessibilidade urbana. O livro começa dando uma visão geral sobre o conceito e indicadores de acessibilidade. Em seguida, ensina como analisar dados espaciais e de redes de transporte para se calcular estimativas de acessibilidade por diferentes modos de transporte e como visualizar esses resultados em mapas e gráficos. O livro também busca familiarizar o leitor com dados de redes de transporte público em formato GTFS e ensina como manipular e visualizar esse tipo de dado. Num dos seus principais capítulos, o livro ensina ainda como os dados e metodologia apresentados nos capítulos anteriores podem ser utilizados para avaliar o impacto de projetos de transporte sobre as condições de acesso a oportunidades da população. Por fim, o livro ensina como baixar, analisar e visualizar estimativas de acessibilidade já calculadas para cidades brasileiras e disponibilizadas pela equipe do Projeto Acesso a Oportunidades (AOP) do Ipea.

O conteúdo do livro foi pensado para as necessidades de trabalho de gestores públicos, além de analistas, alunos e pesquisadores de planejamento e transporte urbano. Por isso, o livro tem caráter prático. Todo material do livro é apresentado com exemplos reproduzíveis e dados abertos utilizando-se programação em R.

Organização do livro

Este livro está dividido em 5 seções:

1. Seção 1: A primeira seção apresenta o conceito de acessibilidade urbana, esclarece a diferença entre acessibilidade e mobilidade urbana, e apresenta os principais indicadores usados para medir a acessibilidade urbana.
2. Seção 2: A segunda seção ensina como calcular estimativas de acessibilidade urbana em R usando os pacotes `r5r` e `accessibility` a partir de dados públicos abertos.
3. Seção 3: A terceira seção apresenta o que é a especificação GTFS de dados de transporte público, e mostra como trabalhar e analisar dados de GTFS usando R.
4. Seção 4: A quarta seção mostra como o conhecimento dos capítulos anteriores pode ser utilizado para se avaliar o impacto de políticas urbanas sobre condições de acessibilidade urbana usando R.

5. Seção 5: Por fim, a quinta seção mostra como fazer download e analisar os dados do projeto Acesso a Oportunidades (AOP) para analisar a distribuição espacial e socioeconômica das condições de acesso a oportunidades nas cidades brasileiras.

Licença de uso

É permitida a reprodução e a exibição para uso educacional ou informativo, desde que respeitado o crédito ao autor original e citada a fonte. Permitida a inclusão da obra em Repositórios ou Portais de Acesso Aberto, desde que fique claro para os usuários os termos de uso da obra e quem é o detentor dos direitos autorais, o Instituto de Pesquisa Econômica Aplicada (Ipea). Proibido o uso comercial ou com finalidades lucrativas em qualquer hipótese. Proibida a criação de obras derivadas. Para imagens estáticas e em movimento (vídeos e audiovisuais), ATENÇÃO: os direitos de imagem foram cedidos apenas para a obra original, formato de distribuição e repositório. Esta licença está baseada em estudos sobre a Lei Brasileira de Direitos Autorais (Lei 9.610/1998).

Como citar o livro

Caso você use o material deste livro, pedimos por gentileza que você cite o livro utilizando a seguinte referência:

- Pereira, Rafael H. M. & Herszenhut, Daniel. (2022) Introdução à acessibilidade urbana um guia prático em R. Instituto de Pesquisa Econômica Aplicada.

Agradecimentos

Este livro foi elaborado pelo Instituto de Pesquisa Econômica Aplicada (Ipea) no âmbito da parceria entre Ipea e a Secretaria de Mobilidade e Desenvolvimento Regional e Urbano (SMDRU) do Ministério do Desenvolvimento Regional (MDR).

i Este livro foi escrito e publicado com o sistema de publicação [Quarto](#). Todo o código utilizado em seu preparo e na sua publicação pode ser encontrado [neste repositório](#).

Part I

SEÇÃO 1: Introdução à acessibilidade urbana

Objetivo: O objetivo desta seção é (1) apresentar o conceito de acessibilidade urbana, esclarecendo a diferença entre acessibilidade e mobilidade; e (2) apresentar uma visão geral dos principais indicadores usados para medir acessibilidade.

Quanto tempo se consegue acessar em menos de uma hora usando transporte público? Quanto tempo se leva para chegar até o posto de saúde ou escola mais próxima da sua casa? As respostas a essas perguntas dependem diretamente das políticas de transporte e de desenvolvimento urbano das cidades. Essas políticas determinam em larga medida a acessibilidade urbana, isto é, a facilidade com a qual pessoas de diferentes grupos sociais e níveis de renda distintos conseguem acessar oportunidades de emprego, serviços de saúde e educação, atividades culturais e de lazer. Assim, essas políticas têm papel-chave para o funcionamento da economia, para a construção de cidades mais sustentáveis e inclusivas e para a redução da desigualdade de acesso a oportunidades.

1 O que é acessibilidade?

1.1 Definição de acessibilidade urbana

Acessibilidade é a facilidade com que as pessoas conseguem alcançar lugares e oportunidades – ou, inversamente, uma característica de lugares e oportunidades em termos de quão facilmente eles podem ser alcançados pela população (Geurs & van Wee, 2004; Neutens et al., 2010).

As condições de acessibilidade são influenciadas tanto pela co-distribuição espacial da população e de atividades econômicas e serviços públicos quanto pela configuração e desempenho das redes de transporte. Nesse sentido, a acessibilidade urbana tem papel fundamental na capacidade das pessoas de se deslocarem para acessar oportunidades, como empregos, escolas, etc.

As condições de acessibilidade urbana são moldadas por três componentes:

- **Infraestrutura:** A facilidade de acessar atividades depende da infra-estrutura e dos serviços de transporte existentes. Isso inclui, por exemplo, a capilaridade da rede de transporte público, a conectividade da rede viária, a existência de corredores de transporte de alta capacidade como trens, metrô etc. Aqui, tanto a eficiência quanto a conectividade espacial e temporal da rede de transporte tem papel chave.
- **Uso do solo:** A facilidade de acessar atividades também depende da co-distribuição espacial de pessoas, áreas residenciais e atividades como escolas, serviços de saúde, áreas de lazer, etc. Esse componente diz respeito à proximidade geográfica entre pessoas e oportunidades.
- **Pessoas:** A facilidade de acessar atividades também é afetada pelas características das pessoas. Fatores como deficiência físicas e cognitivas, idade, gênero, cor, e renda, por exemplo, podem influenciar de maneira importante a velocidade das pessoas se locomoverem, sua capacidade de utilizar determinados modos de transporte, e sua capacidade de circular pela cidade sem medo de algum tipo de violência ou discriminação.

1.2 Por que acessibilidade importa?

O conceito de acessibilidade é central em estudos de transporte por várias razões. Esse conceito articula de maneira mais explícita como políticas de transporte e políticas de desenvolvimento

e uso do solo urbano interagem de maneira a impactar as capacidades das pessoas de se deslocarem nas cidades. O acesso a postos trabalho, serviços de educação e saúde tem papel fundamental para a satisfação das necessidades individuais e sociais, e é uma condição necessária, embora não suficiente, para a expansão da liberdade de escolha das pessoas Lucas et al. (2016). Ademais, a ideia de acessibilidade traz à tona a dimensão espacial da injustiça e desigualdade no acesso a oportunidades, e ajuda a incorporar de maneira explícita a noção de espaço no desenho de políticas destinadas a enfrentar essas injustiças Pereira, Schwanen, and Banister (2017).

O nível de acesso a oportunidades numa cidade é um resultado conjunto da capacidade de as pessoas utilizarem tecnologias de transporte e da integração entre a distribuição geográfica de atividades vis-à-vis a conectividade espacial e temporal da rede de transporte Páez, Scott, and Morency (2012). Assim, a construção de cidades mais inclusivas e sustentáveis passa, em larga medida, por um planejamento integrado entre uso do solo e do sistema de transporte, o que tende a criar maior proximidade entre pessoas e atividades, aumentando a acessibilidade urbana e reduzindo a dependência de modos de transporte motorizados (Banister 2011).

1.3 Diferença entre micro e macro acessibilidade

Para fins de esclarecimento de conceitos, é importante distinguir entre o que nós chamamos de acessibilidade urbana e o uso mais corrente que é feito no português da palavra acessibilidade.

O termo *acessibilidade* é comumente utilizado para se referir a questões relacionadas à normas de design universal, construção e planejamento para inclusão de pessoas com diferentes graus e tipos de deficiência física. Isso é o que nós podemos chamar de microacessibilidade, pois trata na escala micro individual da capacidade de pessoas conseguirem acessar lugares, serviços, produtos etc.

A expressão *acessibilidade urbana*, por sua vez, pode ser entendida como macroacessibilidade, pois trata de uma maneira mais ampla sobre como a capacidade de acessar atividades considerando tanto as capacidades das pessoas utilizarem tecnologias de transporte quanto a distribuição espacial de atividades vis-à-vis a cobertura e conectividade da rede de transporte.

Notadamente, a microacessibilidade é um importante componente de uma noção mais ampla de acessibilidade urbana. Condições de microacessibilidade afetam diretamente a capacidade de pessoas embarcarem e utilizarem diferentes modos de transporte, de se locomoverem com segurança sobre calçadas e atravessar ruas etc.. Ainda, ambas micro e macroacessibilidade têm papel fundamental em moldar as condições de mobilidade urbana da população.

1.4 Diferença entre acessibilidade e mobilidade urbana

O conceito de acessibilidade é diferente, mas complementar ao de mobilidade urbana. Estudos sobre mobilidade urbana costumam olhar para os padrões de viagens que as pessoas efetivamente fazem no seu dia a dia – por exemplo, quantas viagens foram feitas, que modo de transporte as pessoas usam, qual a distância média das viagens, qual o tempo de deslocamento casa-trabalho etc. Informações de mobilidade são tipicamente captadas por meio de pesquisas de origem-destino, ou dados de GPS de telefones celulares, cartões de bilhetagem eletrônica etc. Dados sobre mobilidade urbana trazem informações importantes sobre as condições diárias de transporte e padrões de viagens, que captam aspectos importantes do desempenho econômico e ambiental das cidades e o bem-estar da população.

Tradicionalmente, o planejamento urbano e de transportes tem como foco melhorar a mobilidade urbana Levinson and King (2020). Esse foco na mobilidade tem motivado políticas que priorizam a circulação de automóveis, e visam aumentar a velocidade e a fluidez de trânsito para reduzir congestionamentos (Banister 2011). No entanto, a mobilidade não é um fim em si mesma. Via de regra, as pessoas se deslocam como um meio para acessar as atividades no destino da viagem.

Nesse sentido, tem-se observado um crescente consenso entre pesquisadores e agências de transporte que o objetivo de uma política de transporte é melhorar o acesso da população Bertolini, le Clercq, and Kapoen (2005). Se o que as pessoas querem é acessar atividades, então é possível pensar em formas de planejamento que facilitem as pessoas alcançarem tais atividades sem necessariamente promover o aumento da motorização e da velocidade no trânsito. Isso por ser alcançado, por exemplo, por políticas que promovem maior mix de uso do solo e maior integração entre planejamento de transporte e uso do solo e maior proximidade entre pessoas e atividades.

Quando o foco da política sai da mobilidade e passa a ser melhorar a acessibilidade urbana, abre-se um leque maior de possíveis instrumentos e ações de políticas públicas para promover um desenvolvimento urbano mais integrado e calcado na promoção da sustentabilidade e inclusão social Banister (2011).

2 Indicadores de acessibilidade

Existem diversos indicadores para se medir acessibilidade. Esses indicadores podem ser divididos em dois grandes grupos: indicadores baseados em lugar e indicadores baseados em pessoas (Dijst, de Jong, and van Eck 2002).

2.1 Medidas baseadas em lugares

Medidas baseadas em lugar medem a acessibilidade enquanto uma característica de um determinado local. Por simplificação, esses indicadores assumem que todas as pessoas que se encontram em um mesmo local têm as mesmas condições de acesso às atividades distribuídas pela cidade. Esses indicadores são sensíveis a fatores relacionados à distribuição espacial de atividades e à configuração e desempenho da rede de transporte, mas não levam em consideração as características individuais das pessoas.

Os indicadores desse tipo são os mais amplamente utilizados por agências de transporte e pesquisadores (**boisjoly2017get?**). Isso porque esses indicadores exigem menor quantidade de dados e são consideravelmente mais fáceis de serem calculados. Por este motivo, todo o material deste curso irá focar nesses indicadores de acessibilidade baseados em lugares.

Nós apresentamos abaixo uma rápida descrição de alguns desses indicadores. Note que, em geral, esses indicadores são medidos com base num custo de transporte calculado em termos de tempo de viagem. No entanto, o termo “custo” é utilizado aqui de maneira mais ampla, e pode se referir a outros tipos de custo como a distância de viagem ou seu custo monetário.

2.1.1 Mínimo custo de viagem

O Indicador de mínimo custo de viagem aponta qual o menor custo (por exemplo, em termos de tempo ou distância) de viagem até a oportunidade mais próxima. Ele permite captar, por exemplo, qual o tempo de viagem até o posto de saúde mais próximo. Esse é um dos indicadores mais simples de acessibilidade.

$$A_i = \min(c_{i1}, c_{i2}, \dots, c_{ij}, \dots, c_{i(n-1)}, c_{in}) \iff O_j \geq 1$$

em que A_i é a acessibilidade na origem i , c_{ij} é o custo de deslocamento entre a origem i e o destino j , n é o número total de destinos na área de estudo e O_j é o número de oportunidades no destino j .

Vantagens e desvantagens: Este indicador tem as vantagens de ser fácil de se calcular com pouca exigência de dados, além de ser fácil de comunicar. Duas desvantagens, no entanto, é que ele não capta a quantidade de oportunidades acessíveis e nem aspectos de competição na demanda pela oportunidade. Por exemplo, uma pessoa pode morar muito perto de um hospital, mas essa proximidade pode não garantir um bom acesso aos serviços de saúde se esse for o único hospital da região que fica sobrecarregado com demanda de pacientes.

2.1.2 Medida cumulativa de oportunidades

O indicador de oportunidades cumulativas mede a quantidade de oportunidades que podem ser alcançadas dentro de um tempo máximo de viagem. Por exemplo, este indicador pode ser utilizado para medir a quantidade de empregos acessíveis por transporte público em até 60 minutos, ou a quantidade de escolas acessíveis em até 30 minutos de viagem a pé.

$$A_i = \sum_{j=1}^n O_j \times f(c_{ij})$$

$$f(c_{ij}) = \begin{cases} 1 & \text{se } c_{ij} \leq C \\ 0 & \text{caso contrário} \end{cases}$$

em que A_i é a acessibilidade na origem i , O_j é o número de oportunidades no destino j , n é o número total de destinos na área de estudo, $f(c_{ij})$ é um função binária que assume os valores 0 ou 1, a depender do custo de deslocamento c_{ij} entre a origem i e o destino j , e C é o limite de custo de deslocamento estabelecido.

Vantagens Vs desvantagens: A medida cumulativa de oportunidades também também é fácil de se calcular com pouca exigência de dados, além de ser fácil de comunicar. Isso contribui para tornar este indicador um dos mais utilizados por agências de transporte e de financiamento para analisar acessibilidade (boisjoly2017get?). Entre as suas desvantagens, no entanto, cabe destacar que este indicador não considera a influência da competição sobre oportunidades. Este indicador também exige a escolha de um único ponto de corte como tempo máximo de viagem. Além disso, esta medida assume todas as oportunidades são igualmente desejáveis pelas pessoas, esteja ela a uma distância de 10 ou 40 minutos de viagem, desde que esses tempos de viagem estejam dentro do limite pré-estabelecido.

2.1.3 Medidas gravitacionais

Indicadores gravitacionais de acessibilidade também medem a quantidade de oportunidades acessíveis a partir de um determinado local, mas a contagem de cada oportunidade é gradualmente descontada à medida que o custo da viagem aumenta. Assim, oportunidades mais próximas têm uma importância maior, e o peso de cada oportunidade diminui quanto mais distante ela estiver.

O ritmo de decaimento desse peso em função do custo da viagem é conhecido como função de impedância. Essa função pode ser definida seguindo diversas diferentes fórmulas funcionais. Por exemplo, é possível considerar um decaimento linear. Neste caso, o peso da oportunidade diminui de maneira contínua ao longo do espaço até certo ponto a partir do qual o peso passa a ser zero. Outra opção é considerar uma função negativa exponencial, onde o peso cai muito rapidamente em distâncias mais próximas mas passa a ter uma queda mais suave como pesos igualmente baixos para oportunidades muito distantes.

$$A_i = \sum_{j=1}^n O_j \times f(c_{ij})$$

$$f_{lin}(c_{ij}) = \begin{cases} 1 - c_{ij}/C & \text{se } c_{ij} \leq C \\ 0 & \text{caso contrário} \end{cases}$$

$$f_{exp}(c_{ij}) = e^{-\beta c_{ij}}$$

em que A_i é a acessibilidade na origem i , O_j é o número de oportunidades no destino j , n é o número total de destinos na área de estudo, $f(c_{ij})$ é uma função de decaimento cujo resultado varia com o custo de deslocamento c_{ij} entre a origem i e o destino j , $f_{lin}(c_{ij})$ é a função de decaimento linear, C é o limite de custo de deslocamento estabelecido, $f_{exp}(c_{ij})$ é a função de decaimento exponencial negativa e β é um parâmetro que dita a velocidade de decaimento.

Vantagens Vs desvantagens: A principal vantagem de indicadores gravitacionais é que o desconto do peso das oportunidades pela sua distância reflete de alguma maneira o comportamento de como as pessoas costumam se comportar. Serviços e atividades que gostaríamos de acessar costumam ser mais atrativas quanto mais próximas elas estiverem, tudo mais constante. Este indicador tem ao menos duas desvantagens. A primeira delas é que os valores de acessibilidade estimados são de difícil interpretação pela forma como a contagem de oportunidades é descontada pela distância. Além disso, para que o indicador seja mais representativo do comportamento de viagem das pessoas, a forma funcional e o ritmo de decaimento da função de impedância precisam ser calibradas. Por isso, este indicador também requer a disponibilidade de dados de padrões de viagens disponíveis, por exemplo, a partir de pesquisas de origem destino.

2.1.4 Indicadores de acessibilidade com competição: *floating catchment area*

Em muitos casos, o acesso a oportunidades é afetado não apenas por questões de proximidade e custos de transporte, mas também pela possível competição de muitas pessoas que querem acessar a mesma oportunidade ao mesmo tempo. Isso é muito comum, por exemplo, em casos de acesso a serviços de saúde, escolas e empregos. Uma vaga de emprego só pode ser ocupada por uma pessoa de cada vez, o mesmo vale por exemplo para um leito de UTI ou vaga em uma escola.

Existe uma gama de indicadores de acessibilidade que buscam levar em consideração essa possível competição pelas oportunidades acessíveis. Vários desses indicadores são do tipo *floating catchment area* (áreas de influência flutuantes, em tradução livre). A título de exemplo, esses indicadores tentam levar em consideração como uma mesma pessoa pode potencialmente acessar vários leitos de UTI e, simultaneamente, como cada leito de UTI pode ser acessado por diversas pessoas. Assim, o acesso de uma pessoa ao serviço de leito de UTI é influenciado não apenas por questões de custos de transporte mas também pela disponibilidade de leitos de UTI por pessoas que potencialmente poderiam acessar os mesmos leitos.

Dentro dessa família de indicadores tipo *floating catchment area* (FCA). O indicador mais comumente utilizado é o *2-Step Floating Catchment Area* (2SFCA), proposto originalmente por Luo and Wang (2003). Uma limitação do 2SFCA é que ele contabiliza que uma mesma pessoa pode potencialmente demandar várias oportunidades ao mesmo tempo, e que um mesmo serviço pode ser potencialmente utilizado por várias pessoas ao mesmo tempo. Isso é conhecido como problema de inflação de demanda e de oferta, respectivamente, e pode gerar estimativas enviesadas de acessibilidade (Paez, Higgins, and Vivona 2019). Para lidar com esse problema, Paez et al (2019) propuseram o indicador *Balanced Floating Catchment Area* (“BFCA”), uma das medidas mais novas na família de indicadores tipo *floating catchment area*.

Vantagens Vs desvantagens: Diferentes indicadores desse tipo vão ter pequenas variações em duas vantagens e desvantagens. No entanto, de uma maneira geral, a principal vantagem de indicadores tipo *floating catchment area* é a sua capacidade de incorporar aspectos de competição em medidas de acessibilidade. Uma das desvantagens destes tipos de indicadores é a difícil interpretação e comunicação dos seus resultados.

2.2 Medidas baseadas em pessoas

Indicadores de acessibilidade baseados em pessoas são sensíveis não apenas à distribuição espacial de atividades e a configuração e desempenho da rede de transporte. Esses indicadores mas também levam em consideração como características pessoais (como sexo, idade, deficiência física etc) e até questões como atividades e compromissos pessoais podem afetar a facilidade de uma pessoa acessar determinadas atividades. Esse grupo de medidas inclui, por exemplo, indicadores de acessibilidade baseados em utilidade Miller (2018), indicadores baseados em atividades (Dong et al. 2006) ou medidas de espaço-tempo Neutens et al. (2012).

Embora esses tipos de indicadores sejam mais sofisticados, eles costumam demandar grandes quantidades de dados, incluindo registros de diários de viagem, pesquisas domiciliares tipo origem-destino etc. Por isso, o cálculo desses indicadores é computacionalmente mais intensivo e complexo, o que faz com que esses indicadores sejam menos utilizados Miller (2018).

Part II

SEÇÃO 2: Calculando acessibilidade

Objetivo: O objetivo desta seção é mostrar como calcular estimativas de acessibilidade urbana em R usando os pacotes `r5r` e `accessibility`.

O cálculo dos níveis de acessibilidade em um determinado local compreende duas etapas principais: primeiro, nós precisamos calcular uma matriz de custo, geralmente o tempo de viagem, entre as origens e os destinos de uma determinada cidade ou região considerando um modo de transporte; feito isso, calculamos a acessibilidade em cada ponto de origem considerando os custos de transporte entre cada origem e o número de oportunidades em cada destino. Nesta seção nós aprenderemos mais sobre essas duas etapas usando R, quais dados são necessários para executá-las e quais as vantagens e desvantagens dos diferentes métodos que podem ser usados para isso.

3 Calculando acessibilidade urbana em R

3.1 Cálculo da matriz de tempo de viagem

Como comentado anteriormente, a primeira etapa necessária para calcular os níveis de acessibilidade de uma área urbana é calcular a matriz de custo de viagem entre as diversas origens e destinos que a compõem. Na literatura científica e na prática do planejamento de sistemas de transporte público, esse custo é mais frequentemente representado pelo tempo de viagem que separa dois pontos (El-Geneidy et al. 2016; Venter 2016), embora trabalhos recentes tenham considerado também outros fatores, como o dinheiro necessário para realizar uma viagem e o nível de conforto da viagem entre um ponto e outro (Arbex and Cunha 2020; Herszenhut et al. 2022). Pela prevalência deste tipo de matriz na literatura e na prática, porém, iremos nos focar em matrizes de tempo de viagem.

Atualmente, a forma mais fácil e rápida de gerar uma matriz de tempo de viagem em R é utilizando o pacote `r5r` (Pereira et al. 2021), desenvolvido pela equipe do Projeto Acesso a Oportunidades, do Ipea. O pacote utiliza, por trás dos panos, o *software* de roteamento multimodal de transporte público R5, desenvolvido pela Conveyal¹.

3.1.1 Instalação do `r5r`

A instalação do `r5r` funciona como a instalação de qualquer pacote no R.

```
install.packages("r5r")
```

Além do R, o pacote `r5r` requer também a instalação do Java 11². Se você não sabe qual versão do Java você tem instalada em seu computador, você pode checar essa informação rodando este código no console do R.

```
cat(processx::run("java", args = "--version")$stdout)
```

¹Disponível em <https://github.com/ipeaGIT/gtfstools>.

²Mais detalhes sobre o uso e a sintaxe do `data.table` podem ser lidos em <https://rdatatable.gitlab.io/data.table/index.html>.

```
openjdk 11.0.16 2022-07-19 LTS
OpenJDK Runtime Environment Zulu11.58+15-CA (build 11.0.16+8-LTS)
OpenJDK 64-Bit Server VM Zulu11.58+15-CA (build 11.0.16+8-LTS, mixed mode)
```

3.1.2 Dados necessários

O uso do pacote `r5r` requer os seguintes dados:

- **Rede viária** (obrigatório): um arquivo com a rede viária e de infraestrutura de pedestres do *OpenStreetMap*, em formato `.pbf`;
- **Rede de transporte público** (opcional): um arquivo GTFS descrevendo a rede de transporte público da área de estudo;
- **Topografia** (opcional): um arquivo de dados em *raster* com o modelo digital de elevação em formato `.tif`, caso deseje-se levar em consideração os efeitos da topografia do local sobre os tempos de caminhada.

Aqui estão alguns lugares de onde você pode baixar estes dados:

- OpenStreetMap
 - [osmextract](#), pacote de R
 - [geofabrik](#), website
 - [hot export tool](#), website
 - [BBBike.org](#), website
- GTFS
 - [tidytransit](#), pacote de R
 - [transitland](#), website
 - No capítulo X desde livro (tabela xx) nós indicamos também onde baixar os dados de GTFS de algumas cidades brasileiras que compartilham seus dados publicamente.
- Topografia
 - [elevatr](#), pacote de R
 - [Nasa's SRTMGL1](#), website

Os arquivos destes dados devem ser salvos em uma mesma pasta que, preferencialmente, não contenha nenhum outro arquivo. Como se verá adiante, o `r5r` combina todos os dados salvos nesta pasta para criar uma rede de transporte multimodal que será utilizada para simulações de roteamento e cálculo das matrizes de viagem. Note que é possível ter mais de um arquivo GTFS na mesma pasta, nesse caso o `r5r` irá considerar as redes de transporte públicos de todos *feeds*. No entanto, a pasta deve conter um único arquivo de rede viária `.pbf`. Assumindo que os scripts de R estarão em uma pasta chamada `R`, a organização dos arquivos deverá seguir o esquema abaixo:

```

/tmp/RtmpKJGCVd/projeto_acessibilidade
+-- R
|   +-- script1.R
|   \-- script2.R
\-- r5
    +-- rede_transporte_publico.zip
    +-- rede_viaria.osm.pbf
    \-- topografia.tif

```

Para ilustrar as funcionalidades do `r5r`, nós vamos usar uma pequena amostra de dados para a cidade de Porto Alegre (Brasil). Esses dados estão disponíveis dentro do próprio pacote `r5r` na pasta `system.file("extdata/poa", package = "r5r")`:

```

/home/runner/work/intro_access_book/intro_access_book/renv/library/R-4.2/x86_64-pc-linux-gnu
+-- poa.zip
+-- poa_elevation.tif
+-- poa_hexgrid.csv
+-- poa_osm.pbf
\-- poa_points_of_interest.csv

```

Esta pasta possui quatro arquivos que vamos usar agora:

- A rede viária do OpenStreetMap: `poa_osm.pbf`
- Dois feeds de GTFS das redes de ônibus e de trens: `poa_eptc.zip` e `poa_trensubr.zip`
- O dado de topografia: `poa_elevation.tif`
- Um arquivo `poa_hexgrid.csv` com coordenadas geográficas dos centróides de uma grade hexagonal regular cobrindo toda a área da amostra e com informações sobre o tamanho da população residente e o número de oportunidades (empregos, escolas e hospitais) em cada hexágono. Esses pontos serão utilizados como os pontos de origem e destino no cálculo da matriz de tempo de viagem.

3.1.3 Calculando a matriz de tempo de viagens

Antes de calcular a matriz de tempo de viagem, precisamos aumentar a memória disponível para o Java. Isto é necessário porque, por padrão, o R aloca apenas 512MB de memória para processos Java, o que não é suficiente para grandes consultas usando `r5r`. Para aumentar a memória disponível para 2GB, por exemplo, precisamos definir um valor para o parâmetro `java.parameters` no início do script antes de carregar as bibliotecas do R:

```
options(java.parameters = "-Xmx2G")
```

Pronto, agora vamos carregar as bibliotecas que vamos utilizar neste capítulo.

```
library(r5r)
library(accessibility)
library(sf)
library(data.table)
library(ggplot2)
library(aopdata)
```

Feito isso, o cálculo de uma matriz de tempo de viagens da sua área de estudo pode ser feito em dois passos. O primeiro passo é gerar a rede de transporte multimodal que será utilizada no roteamento. Para isso, nós utilizamos a função `setup_r5()`. Esta função baixa o *software* de roteamento R5 e o utiliza para criar a rede. A função `setup_r5()` recebe como *input* o caminho da pasta onde você armazenou seus dados. Além da função salvar na pasta alguns arquivos necessários para o roteamento, ela também retorna uma conexão com o R5, que neste exemplo nós chamamos de `r5r_core`, e que será utilizada no cálculo da matriz de tempo de viagem.

```
path <- system.file("extdata/poa", package = "r5r")

r5r_core <- setup_r5(path, use_elevation = TRUE, verbose = FALSE)

fs::dir_tree(path)
```

```
/home/runner/work/_temp/renv/cache/v5/R-4.2/x86_64-pc-linux-gnu/r5r/0.7.1/61db001154cd833466
+-- network.dat
+-- poa.zip
+-- poa_elevation.tif
+-- poa_hexgrid.csv
+-- poa_osm.pbf
+-- poa_osm.pbf.mapdb
+-- poa_osm.pbf.mapdb.p
\-- poa_points_of_interest.csv
```

O passo final é usar a função para o cálculo da matriz de tempo de viagem, apropriadamente chamada de `travel_time_matrix()`. Como *inputs* básicos, a função recebe a conexão com o R5 criada acima, pontos de origem e destino em formato de `data.frame` com as colunas `id`, `lon` e `lat`, o modo de transporte a ser utilizado, o horário de partida, o tempo máximo de caminhada permitido da origem até o embarque no transporte público e do desembarque até o destino, e o tempo máximo de viagem a ser considerado.

```
points <- fread(file.path(path, "poa_hexgrid.csv"))
```

```
ttm <- travel_time_matrix(
  r5r_core,
  origins = points,
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  departure_datetime = as.POSIXct(
    "13-05-2019 14:00:00",
    format = "%d-%m-%Y %H:%M:%S"
  ),
  max_walk_dist = 800,
  max_trip_duration = 120,
  verbose = FALSE,
  progress = FALSE
)

ttm
```

	fromId	toId	travel_time
1:	89a901291abffff	89a901291abffff	1
2:	89a901291abffff	89a901295b7ffff	41
3:	89a901291abffff	89a9012809bffff	43
4:	89a901291abffff	89a901285cfffff	35
5:	89a901291abffff	89a90e934d7ffff	71

1169147:	89a90166da7ffff	89a90129133ffff	58
1169148:	89a90166da7ffff	89a9012ac43ffff	93
1169149:	89a90166da7ffff	89a90129a47ffff	19
1169150:	89a90166da7ffff	89a90128883ffff	65
1169151:	89a90166da7ffff	89a90166da7ffff	0

Diversos outros *inputs* podem ser passados para o cálculo da matriz, como a velocidade de caminhada e o número máximo de pernas de transporte público permitido, entre outros. Para mais informações sobre cada um dos parâmetros, por favor consulte a documentação da função disponível no pacote ou no site do **r5r** [aqui](#).

Na prática, o que a função `travel_time_matrix()` faz é encontrar qual a rota de viagem mais rápida partindo de cada ponto de origem para todos os possíveis pontos de destino considerando o modo de viagem, horário de partida e demais parâmetros passados pelo usuário. Para isso, o **r5r** considera tempos de viagem de porta-a-porta. No caso de uma viagem por transporte público, por exemplo, o tempo total de viagem inclui a) o tempo de caminhada até a parada de transporte público; b) o tempo de espera pelo veículo na parada; c) o tempo de deslocamento dentro do veículo; e d) o tempo de viagem a pé da parada de desembarque até o destino. Em

casos em que mais de uma rota de transporte público é utilizada, o **r5r** também contabiliza o tempo gasto nas conexões, considerando a caminhada entre paradas e o tempo de espera pelo próximo veículo.

i A função `travel_time_matrix()` utiliza uma extensão do algoritmo de roteamento RAPTOR (Conway, Byrd, and van der Linden 2017), o que torna o R5 extremamente rápido. A depender da quantidade de pares de origem-destino, o **r5r** pode ser entre 6 e 200 vezes mais rápido do que *softwares* alternativos para calcular uma matriz de tempo de viagem (Higgins et al. 2022).

3.2 Cálculo de acessibilidade

Após calculada a matriz de tempo de viagem entre as origens e os destinos da área de estudo, nós precisamos utilizá-la para calcular os níveis de acessibilidade do local. Para isso, nós utilizaremos o pacote **accessibility**, também desenvolvido pela equipe do Projeto Acesso a Oportunidades, que contém diversas funções para vários indicadores de acessibilidade.

Como *input* básico, todas as funções requerem uma matriz de custo pré-calculada (no nosso caso, a matriz de tempo de viagem calculada na seção anterior) e dados de uso do solo, como o número de determinados tipos de oportunidades em cada ponto da área de estudo, por exemplo.

Medida cumulativa de acesso a oportunidades

O exemplo abaixo mostra uma simples aplicação da função `cumulative_cutoff()`, utilizada para calcular o número de oportunidades que pode ser alcançado em um determinado limite de custo de viagem. No exemplo abaixo, nós calculamos o número de postos de trabalho que podem ser alcançados em até 30 minutos de viagem a partir de cada origem presente em nossa matriz de tempo de viagem.

```
setnames(ttm, c("fromId", "toId"), c("from_id", "to_id"))

cumulative_access <- cumulative_cutoff(
  ttm,
  points,
  opportunity = "schools",
  travel_cost = "travel_time",
  cutoff = 30
)

cumulative_access
```



```

              id schools
1: 89a901291abffff    19
2: 89a9012a3cffff    0
3: 89a901295b7ffff    15
4: 89a901284a3ffff    0
5: 89a9012809bffff    20
---
1223: 89a90129133ffff    4
1224: 89a9012ac43ffff    9
1225: 89a90129a47ffff    12
1226: 89a90128883ffff    31
1227: 89a90166da7ffff    15

```

Mínimo custo de viagem

A função `cost_to_closest()`, por sua vez, calcula o mínimo custo de viagem necessário para alcançar um determinado número de oportunidades. O código abaixo, por exemplo, calcula o tempo de viagem mínimo para alcançar o hospital mais próximo a partir de cada origem.

```

min_cost <- cost_to_closest(
  ttm,
  points,
  opportunity = "schools",
  travel_cost = "travel_time"
)

```

Warning in ``[.data.table` (filled_access_df, is.na(get(access_col))),`
``:=` (eval(access_col, : inf (type 'double') at RHS position 1 truncated`
 (precision lost) when assigning to type 'integer' (column 2 named 'min_cost')

```

min_cost

              id travel_time
1: 89a9012124ffff    0
2: 89a9012126bffff    16
3: 89a9012127bffff    11
4: 89a90128003ffff    7
5: 89a90128007ffff    20
---
1223: 89a90e934cbffff    13
1224: 89a90e934cffff    9

```

1225: 89a90e934d3ffff	7
1226: 89a90e934d7ffff	0
1227: 89a90e934dbffff	23

Medidas gravitacionais de acessibilidade

A função `gravity()` calcula medidas gravitacionais de acessibilidade - ou seja, aquelas nas quais o peso de cada oportunidade diminui gradualmente com o aumento do custo de viagem. Existem, no entanto, uma gama de diferentes tipos de funções de decaimento que podem ser utilizadas, como funções de decaimento exponenciais negativas, de potências inversas, entre outras. Por isso, esta função recebe um *input* adicional: a função de decaimento a ser utilizada no cálculo. O exemplo abaixo apresenta o cálculo de acessibilidade a estabelecimentos de educação usando uma medida gravitacional exponencial negativa com parâmetro de decaimento igual a 0.2.

```
negative_exp_access <- gravity(
  ttm,
  points,
  opportunity = "schools",
  travel_cost = "travel_time",
  decay_function = decay_exponential(0.2)
)
```

```
negative_exp_access
```

	id	schools
1:	89a901291abffff	0.35892873
2:	89a9012a3cffff	0.00000000
3:	89a901295b7ffff	0.52061269
4:	89a901284a3ffff	0.00000000
5:	89a9012809bffff	0.44769080

1223:	89a90129133ffff	0.02885676
1224:	89a9012ac43ffff	0.09138463
1225:	89a90129a47ffff	0.35857154
1226:	89a90128883ffff	1.86366449
1227:	89a90166da7ffff	0.40513107

Indicadores de acessibilidade com competição

Por fim, a função `floating_catchment_area()` calcula níveis de acessibilidade levando em consideração a competição por oportunidades usando diferentes indicadores do tipo *floating*

catchment area. Como diversos métodos de FCA podem ser utilizados, a função requer que o método desejado seja explicitamente assinalado. E, assim como a função de acessibilidade gravitacional, a função de decaimento utilizada também deve ser definida pelo usuário. O código a seguir mostra um exemplo de cálculo de acessibilidade a oportunidades de emprego usando o método BFCA (Paez, Higgins, and Vivona 2019), levando em consideração os efeitos de competição entre a população como um todo e uma função de decaimento exponencial com parâmetro de decaimento igual a 2.

```
bfca_access <- floating_catchment_area(  
  ttm,  
  points,  
  opportunity = "schools",  
  travel_cost = "travel_time",  
  demand = "population",  
  method = "bfca",  
  decay_function = decay_exponential(0.05)  
)
```

```
bfca_access
```

```
      id      schools  
1: 89a901291abffff 0.0002574442  
2: 89a9012a3cffff 0.0000000000  
3: 89a901295b7ffff 0.0002091911  
4: 89a901284a3ffff 0.0000000000  
5: 89a9012809bffff 0.0002276971  
---  
1223: 89a90129133ffff 0.0001352617  
1224: 89a9012ac43ffff 0.0001416331  
1225: 89a90129a47ffff 0.0001777023  
1226: 89a90128883ffff 0.0002372291  
1227: 89a90166da7ffff 0.0001957669
```

As funções apresentadas nesta seção podem receber também outros *inputs* não explicitamente mencionados aqui. Para mais informações sobre cada um dos parâmetros, por favor consulte a documentação do pacote `accessibility` no seu [site](#).

3.3 Cálculo de acessibilidade com o r5r

Ao longo das duas seções anteriores, nós mostramos como calcular níveis de acessibilidade passo-a-passo. Para fins didáticos, é importante entender que o cálculo de estimativas de aces-

sibilidade tem como primeiro passo a geração de uma matriz de custos de viagens a partir de simulações de roteamento, que posteriormente é utilizada para estimar níveis de acessibilidade.

No entanto, o `r5r` inclui também uma função chamada `accessibility()` que faz todo esse processamento com uma única chamada. De forma parecida com a função de cálculo de matriz de tempo de viagem, a função de acessibilidade também recebe como *inputs* uma conexão com o R5, as origens, os destinos, os modos de transporte, o tempo de partida, entre outros. Adicionalmente, devem ser listadas também quais oportunidades devem ser consideradas e a função de decaimento que deve ser utilizada (bem como o valor do limite de custo e de parâmetro de decaimento). O exemplo abaixo mostra uma aplicação desta função.

```
r5r_access <- accessibility(
  r5r_core,
  origins = points,
  destinations = points,
  opportunities_colname = "schools",
  decay_function = "step",
  cutoffs = 30,
  mode = c("WALK", "TRANSIT"),
  departure_datetime = as.POSIXct(
    "13-05-2019 14:00:00",
    format = "%d-%m-%Y %H:%M:%S"
  ),
  max_walk_dist = 800,
  max_trip_duration = 120,
  verbose = FALSE,
  progress = FALSE
)
```

```
r5r_access
```

	from_id	percentile	cutoff	accessibility
1:	89a901291abffff	50	30	17
2:	89a9012a3cffff	50	30	0
3:	89a901295b7ffff	50	30	13
4:	89a901284a3ffff	50	30	0
5:	89a9012809bffff	50	30	17

1223:	89a90129133ffff	50	30	1
1224:	89a9012ac43ffff	50	30	8
1225:	89a90129a47ffff	50	30	10

1226: 89a90128883ffff	50	30	30
1227: 89a90166da7ffff	50	30	14

Como podemos observar, o resultado desta função são os níveis de acessibilidade já calculados. A informação “intermediária” de tempo de viagem entre origens e destinos, consequentemente, não fica disponível ao usuário com o uso da função. Ainda assim, este fluxo de trabalho pode ser uma boa alternativa para pessoas que estejam interessados puramente nos níveis de acessibilidade e não dependam do tempo de viagem em suas análises.

3.4 Análises de acessibilidade

Calculados os níveis de acessibilidade, procedemos então para sua análise. Existe uma grande variedade de análises que podem ser feitas usando esses dados, por exemplo de diagnóstico das condições de acessibilidade urbana de diferentes bairros, pesquisas sobre desigualdades de acesso a oportunidades entre diferentes grupos sociais, análises sobre exclusão social e *accessibility poverty* (insuficiência de acessibilidade, tradução livre), etc. Nesta seção, no entanto, nos restringiremos a apresentar duas análises relativamente simples e de fácil comunicação: a distribuição espacial da acessibilidade e sua distribuição entre diferentes grupos de renda.

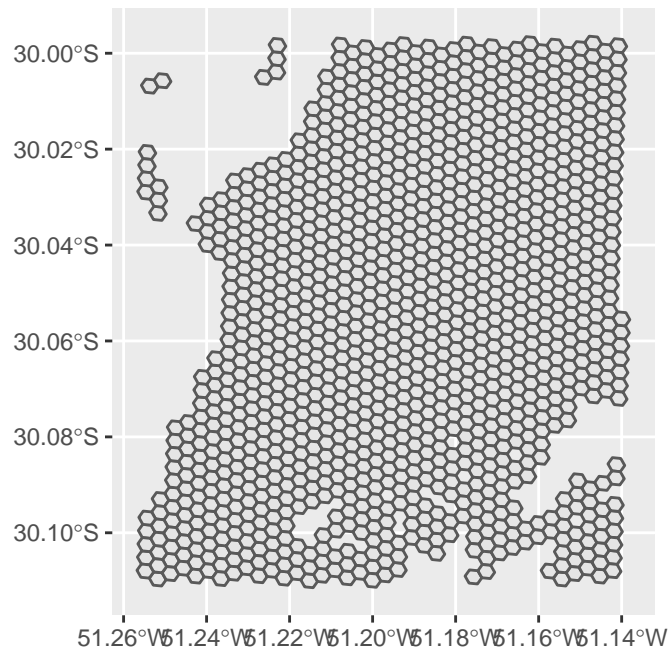
Distribuição espacial de acessibilidade urbana

Para compreendermos a distribuição espacial da acessibilidade urbana de uma determinada cidade ou região, primeiro precisamos obter as informações espaciais dos pontos que foram utilizados como origens e destinos no cálculo da matriz. Os pontos que nós usamos nos exemplos anteriores, por exemplo, correspondem a células de uma grade hexagonal baseadas no índice H3, desenvolvido pela Uber (Brodsky 2018). O código e o mapa a seguir mostram a distribuição desses hexágonos na nossa área de estudo.

```
poa_grid <- read_grid("Porto Alegre")

poa_grid <- poa_grid[poa_grid$id_hex %in% points$id, ]

ggplot(poa_grid) + geom_sf()
```

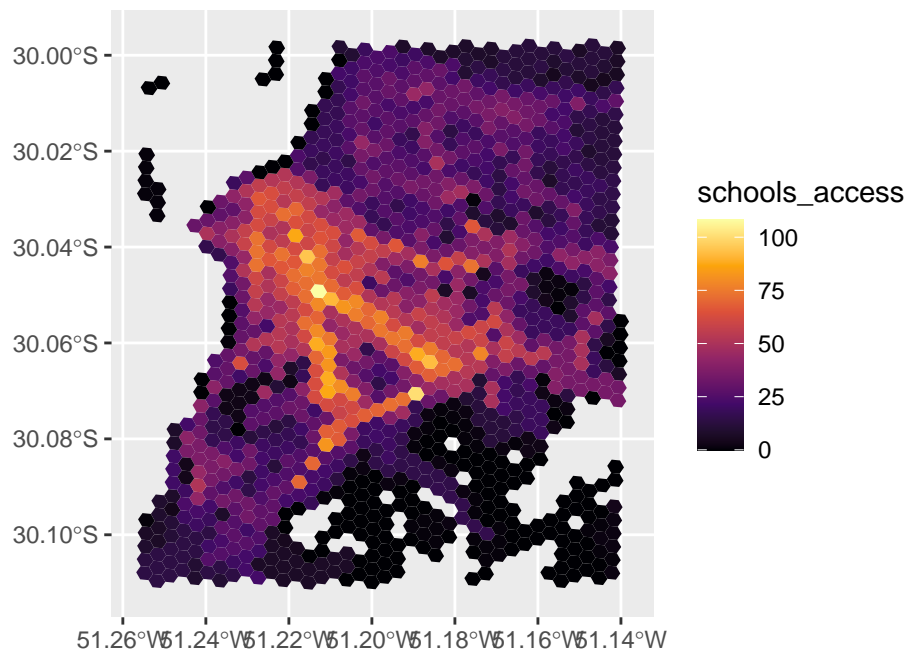


Para visualizarmos os nossos dados de acessibilidade espacialmente, portanto, nós precisamos unir a nossa tabela com estimativas de acessibilidade com a tabela que contém os dados espaciais da grade usando a coluna de `id` dos hexágonos como nossa coluna-chave. Os comandos usados para isso, bem como o resultado da operação em formato de imagem, podem ser vistos no código e no mapa a seguir.

```
setDT(poa_grid)
poa_grid[cumulative_access, on = c(id_hex = "id"), schools_access := i.schools]

poa_grid_sf <- st_sf(poa_grid)

ggplot(poa_grid_sf) +
  geom_sf(aes(fill = schools_access), color = NA) +
  scale_fill_viridis_c(option = "inferno")
```



Como podemos ver, os níveis de acessibilidade tendem a se concentrar de forma mais acentuada no centro da cidade, onde existe maior concentração de empregos, e próximos aos grandes corredores de transportes da cidade. Pessoas que moram mais perto desses corredores tendem a gastar menos tempo no acesso a suas estações e conseguem acessar locais distantes relativamente rápido, por terem fácil acesso a modos de alta capacidade e velocidade. Pessoas que moram mais longe desses corredores, por sua vez, dependem de modos de menor velocidade operacional (como os ônibus municipais, por exemplo) e precisam gastar mais tempo para alcançar os corredores de média e alta capacidade. Como consequência, seus níveis de acessibilidade tendem a ser menores.

Distribuição socioeconômica de acessibilidade urbana

O mapa acima, embora revelador quanto aos locais em que estão dispostas as maiores concentrações de acessibilidade, nada diz sobre quais são os grupos socioeconômicos que possuem os maiores potenciais de acesso a oportunidades na região. Para isso, nós precisamos cruzar informações demográficas e econômicas de cada um dos nossos pontos de origem com os dados de acessibilidade previamente calculados. No exemplo abaixo, nós juntamos aos dados de acessibilidade a informação do decil de renda de cada uma das origens. Assim, nós conseguimos identificar se um hexágono é de baixa, média ou alta renda.

Os dados com as características socioeconômicas e perfil de renda que utilizamos no exemplo vêm do censo demográfico brasileiro de 2010, e foram agregados na grade espacial de dos hexágonos por ([pereira2022usosolo?](#)). Aqui, nós acessamos os dados diretamente de dentro do R usando o pacote `aopdata`, que será apresentado em mais detalhes no capítulo Xx.

```

poa_population <- read_population("Porto Alegre", showProgress = FALSE)

poa_grid[
  poa_population,
  on = "id_hex",
  `:=`(
    pop_count = i.P001,
    income_decile = i.R003
  )
]

poa_grid[, .(id_hex, schools_access, pop_count, income_decile)]

```

	id_hex	schools_access	pop_count	income_decile
1:	89a9012124fffff	14	733	9
2:	89a9012126bffff	13	355	9
3:	89a9012127bffff	14	996	10
4:	89a90128003ffff	34	1742	4
5:	89a90128007ffff	15	477	5

1218:	89a90e934cbffff	7	118	4
1219:	89a90e934cfffff	12	518	6
1220:	89a90e934d3ffff	5	846	6
1221:	89a90e934d7ffff	12	1615	7
1222:	89a90e934dbffff	6	0	NA

Como vocês podem perceber, nós também trouxemos a informação de contagem populacional em cada origem. Isto se dá porque nós iremos calcular a distribuição da acessibilidade de cada um dos decis de renda. Para isso, portanto, nós precisamos ponderar o nível de acessibilidade de cada origem pela quantidade de pessoas que residem ali. Desta forma, nós teremos a distribuição da acessibilidade das pessoas localizadas em origens de um determinado decil de renda. Caso optássemos por não ponderar, no entanto, nós teríamos a distribuição de acessibilidade não das pessoas localizadas em cada hexágono, mas dos hexágonos em si. Como em nossa análise nós nos importamos com as pessoas, e não com as células espaciais, precisamos fazer a ponderação. Nós podemos visualizar a distribuição de acessibilidade de cada decil usando um boxplot, como feito a seguir.

```

poa_grid[, income_decile := as.factor(income_decile)]

ggplot(poa_grid[!is.na(income_decile)]) +
  geom_boxplot(

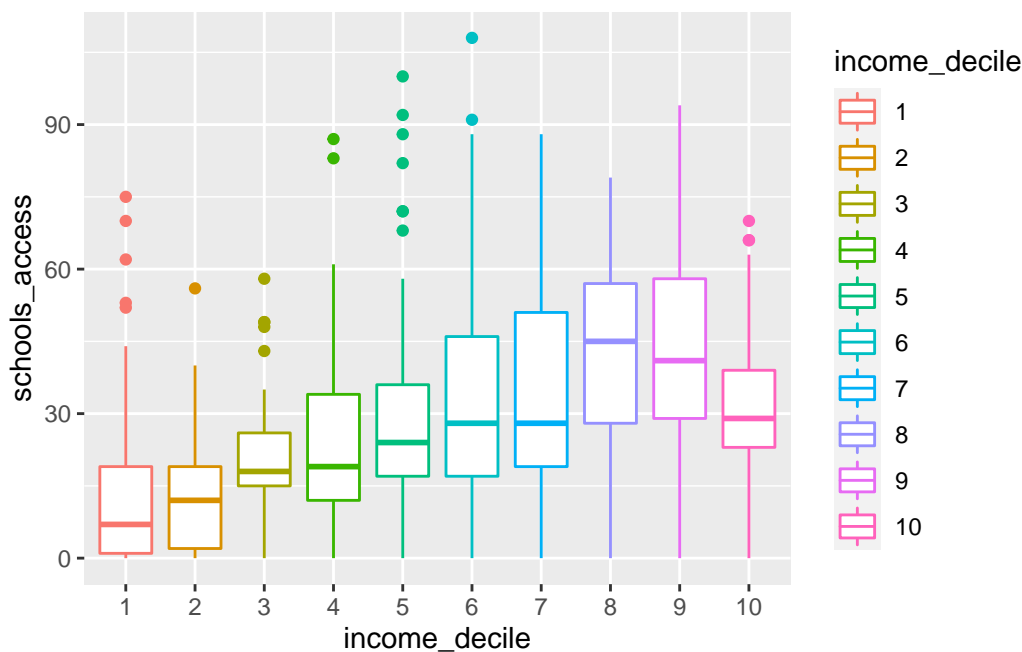
```



```

aes(
  x = income_decile,
  y = schools_access,
  color = income_decile,
  weight = pop_count
)
)

```



O gráfico é muito claro em seu conteúdo: pessoas de mais baixa renda tendem a ter níveis de acessibilidade muito menores do que as de alta renda. Isso é um padrão comum em praticamente todas as cidades brasileiras e em diversas cidades do mundo ([pereira2019desigualdades?](#)). Isto ocorre, em larga medida, devido à localização espacial das comunidades de baixa e alta renda no território: os mais ricos costumam morar em áreas mais valorizadas, próximas das grandes concentração de empregos (e oportunidades de educação, saúde, lazer, etc., também) e com maior oferta de transporte público de média e alta capacidade. Os mais pobres, por sua vez, tendem a morar em locais mais afastados, onde o valor da terra é menor. Consequentemente, tendem também a se afastar das grandes concentrações de oportunidades. Junta-se a isso o fato de, na maior parte dos casos, a oferta de serviços de transporte público de média e alta capacidade ser menor em locais com maior concentração de pessoas de baixa renda. Como consequência, seus níveis de acessibilidade são, em média, muito menores do que os dos mais ricos, como o gráfico deixa claro.

Part III

SEÇÃO 3: Dados de transporte público

Objetivo: O objetivo desta seção é (1) apresentar o que é a especificação de dados de transporte público em formato GTFS; e (2) apresentar como trabalhar e analisar dados de GTFS usando R.

Dados de transporte público são peças fundamentais no planejamento de transportes em geral, e em análises de acessibilidade em particular. Para serem usados de forma que se tenha segurança no resultado das análises, esses dados precisam ser confiáveis e de simples inspeção e interpretação.

Tentando satisfazer esses critérios, cada vez mais agências de transporte público, tomadores de decisão e pesquisadores têm buscado utilizar dados estruturados conforme especificações abertas e colaborativas - ou seja, cujo formato seja decidido por uma comunidade de atores interessados, incluindo partes que produzem esses dados (agências de transporte público, por exemplo) e que os consomem (pesquisadores, desenvolvedores de ferramentas de planejamento, etc.). Embora uma especificação aberta não necessariamente resolva o problema da qualidade e da confiabilidade dos dados por ela descritos, ela traz várias vantagens. O uso de um formato padrão de dados para transporte público permite o desenvolvimento e compartilhamento de ferramentas e programas computacionais para análise desses dados. Assim, um programa desenvolvido por agências de transporte para qualquer cidade no Brasil, Estados Unidos ou Japão pode ser facilmente utilizado para quaisquer cidades do mundo que também organizam seus dados naquele formato padrão. Além disso, quanto mais amplamente utilizado é esse formato padrão de dados, maior tende a ser tanto a confiabilidade da especificação em si quanto a facilidade inspeção dos dados e de sua interpretação, visto que múltiplos atores detêm o conhecimento necessário para tal.

A especificação de dados aberta e colaborativa mais amplamente utilizada no contexto do planejamento de transporte público é o formato GTFS, sigla para *General Transit Feed Specification* (Especificação Geral de Redes de Transporte Público, em tradução livre). Seus usos abrangem tanto o planejamento quanto a operação de sistemas de transporte público. Como visto no capítulo XX, os dados de GTFS também são uma peça fundamental para calcular estimativas de acessibilidade urbana por transporte público. Nesta seção nós iremos aprender em mais detalhe o que são os dados GTFS, como eles são estruturados e como trabalhar com esses dados no R.

4 Dados GTFS

O formato GTFS é uma especificação aberta e colaborativa que visa descrever os principais componentes de uma rede de transporte público. Atualmente, dados GTFS podem ser divididos em duas grandes categorias:

- *GTFS Schedule*, ou *GTFS Static*, que contém o cronograma estático de linhas de transporte público e informações espaciais sobre o itinerário de cada linha e suas paradas;
- *GTFS Realtime*, que contém informações de localização de veículos em tempo real e alertas de possíveis atrasos, de mudanças de percurso e de eventos que possam interferir no cronograma planejado.

Ao longo desta seção, nós focaremos no **GTFS Schedule**. Clique [aqui](#) para mais informações sobre o GTFS Realtime.

Por ser uma especificação aberta e colaborativa, o formato GTFS tenta abarcar em sua definição um grande número de usos distintos que agências de transporte e desenvolvedores de ferramentas possam dar a ele. No entanto, agências e *softwares* podem ainda assim depender de informações que não constem na especificação oficial. Surgem, dessa forma, [extensões](#) da especificação. Algumas dessas extensões podem eventualmente se tornar parte da especificação oficial, caso isto seja aceito pela comunidade de usuários do GTFS. Nesta seção nós focaremos em um subconjunto de informações presentes no formato GTFS Schedule “puro”, e, portanto, não cobriremos suas extensões.

4.1 Estrutura dos arquivos de GTFS

Usualmente, refere-se a um arquivo no formato GTFS Schedule (daqui em diante chamado apenas de GTFS) como *feed*. Neste livro, nós utilizaremos os termos *feed* e arquivo GTFS como sinônimos.

Um *feed* é nada mais do que um arquivo comprimido em formato **.zip** que contém um conjunto de tabelas salvas em formato **.txt** com algumas informações sobre a rede de transporte público (e.g. localização das paradas, frequências das viagens, traçado das rotas etc). Como em uma base de dados relacional, as tabelas de um *feed* têm algumas colunas-chave que permitem vincular os dados rotas, viagens, tabelas de horários etc. O esquema geral do GTFS é mostrado na Figure 4.1, mostrando apenas algumas das principais tabelas e com as colunas-chave destacadas como os pontos finais das setas.

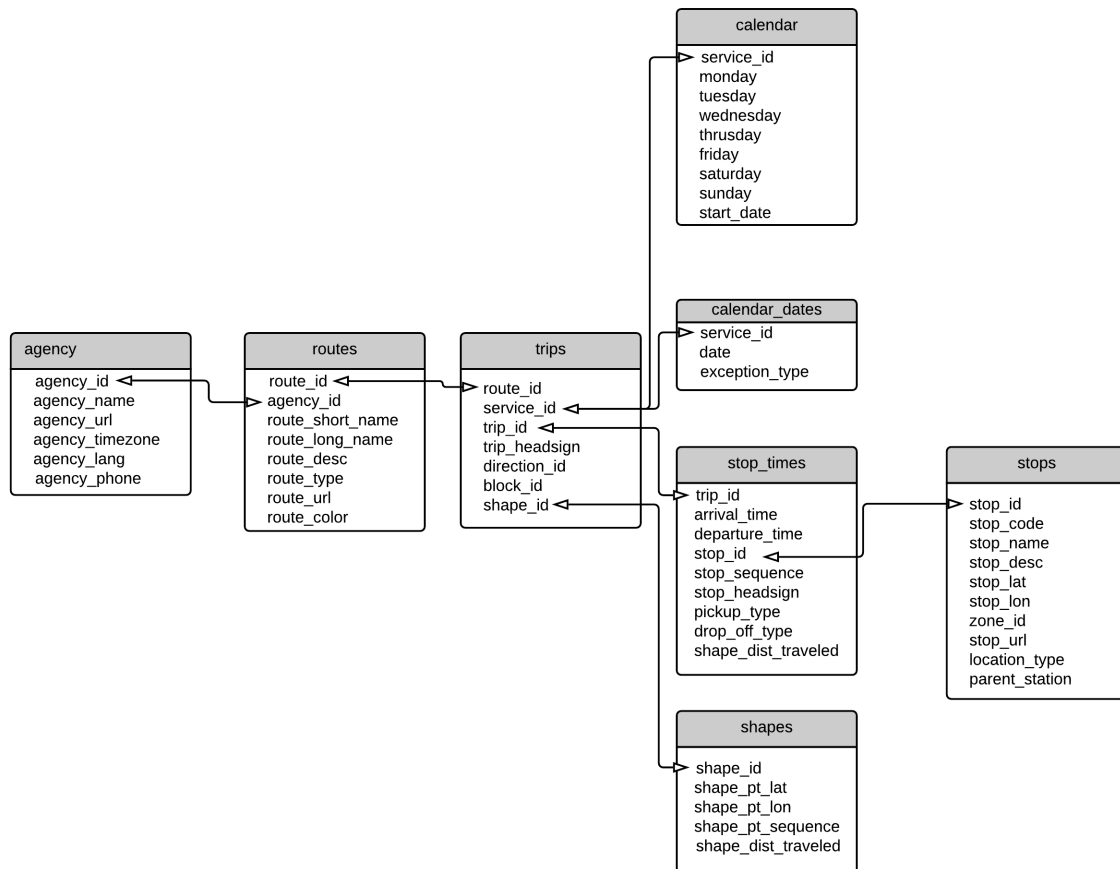


Figure 4.1: Esquema do formato GTFS

Ao todo, 22 tabelas compõem o formato GTFS¹. Nem todas, no entanto, são obrigatórias para que um feed seja considerado válido - sendo consideradas, portanto, opcionais. A especificação classifica cada tabela conforme sua obrigatoriedade em três possíveis categorias: obrigatórias, opcionais e condicionalmente obrigatórias (quando a obrigatoriedade de uma tabela depende da existência de uma determinada tabela, coluna ou valor). Para fins de simplicidade, neste livro nós consideraremos apenas as duas primeiras categorias e faremos comentários quanto à obrigatoriedade de cada tabela quando apropriado. Desta forma, ficam assim classificadas as tabelas:

- Obrigatórias: `agency.txt`, `stops.txt`, `routes.txt`, `trips.txt`, `stop_times.txt`, `calendar.txt`.
- Opcionais: `calendar_dates.txt`, `fare_attributes.txt`, `fare_rules.txt`, `fare_products.txt`, `fare_leg_rules.txt`, `fare_transfer_rules.txt`, `areas.txt`, `stop_areas.txt`, `shapes.txt`, `frequencies.txt`, `transfers.txt`, `pathways.txt`, `levels.txt`, `translations.txt`, `feed_info.txt`, `attributions.txt`.

Ao longo desta seção, nós aprenderemos a estrutura básica de um arquivo GTFS e das tabelas que o compõem. Portanto, vamos olhar apenas para as tabelas obrigatórias e para as tabelas opcionais mais frequentemente utilizadas por produtores e consumidores desses arquivos. Para mais informações sobre as tabelas e as colunas não abordadas nesta seção, por favor verifique a [especificação oficial](#).

Nesta demonstração, nós utilizaremos um subconjunto de dados provenientes do *feed* da cidade de São Paulo criado pela SPTrans² e baixado em outubro de 2019. O *feed* contém as seis tabelas obrigatórias e mais duas tabelas opcionais bastante utilizadas, `shapes.txt` e `frequencies.txt`, o que permite uma boa visão geral sobre o formato GTFS.

4.1.1 `agency.txt`

Arquivo utilizado para descrever as operadoras de transporte que atuam no sistema descrito no arquivo GTFS. Embora o termo *agency* (agência) seja usado em lugar de *operators* (operadoras), por exemplo, fica a cargo do produtor do *feed* definir quais instituições serão listadas na tabela.

Por exemplo: múltiplas concessionárias de ônibus atuam em um determinado local, mas todo o planejamento de cronograma e de tarifa é realizado por uma instituição, em geral uma secretaria de transporte ou empresa pública específica. Esta instituição é também entendida pelos usuários do sistema como a operadora, de fato. Neste caso, devemos listar a instituição responsável pelo planejamento na tabela.

¹Disponível em <https://github.com/ipeaGIT/gtfstools>.

²Mais detalhes sobre o uso e a sintaxe do `data.table` podem ser lidos em <https://rdatatable.gitlab.io/data.table/index.html>.

Agora, imagine um sistema em que a agência de transporte público local transfere a responsabilidade da operação de um sistema multimodal a diversas empresas, por meio de concessões. Cada uma dessas empresas é responsável pelo planejamento de cronogramas e tarifas dos modos que operam, desde que sejam seguidos determinados parâmetros pré-estabelecidos em contrato. Neste caso, devemos listar as operadoras (concessionárias) na tabela, e não a agência de transporte público em si.

A Table 4.1 mostra o arquivo `agency.txt` do *feed* da SPTrans. Como vocês podem ver, os responsáveis pelo *feed* optaram por listar a própria empresa no arquivo, e não as concessionárias que operam os ônibus e o metrô da cidade.

Table 4.1: Exemplo de arquivo `agency.txt`

agency_id	agency_name	agency_url	agency_timezone	agency_lang
1	SPTRANS	http://www.sptrans.com.br/?versao=011019	America/Sao_Paulo	pt

É necessário notar que, embora estejamos apresentando o `agency.txt` em formato de tabela, o arquivo deve ser formatado como se fosse salvo em formato `.csv`. Ou seja, os valores de cada célula da tabela devem ser separados por vírgulas, e cada linha da tabela deve constar em uma linha no arquivo. A tabela acima, por exemplo, é definida da seguinte forma:

```
agency_id,agency_name,agency_url,agency_timezone,agency_lang
1,SPTRANS,http://www.sptrans.com.br/?versao=011019,America/Sao_Paulo,pt
```

Por uma questão de comunicação e interpretação dos dados, ao longo desta seção sempre apresentaremos os exemplos em formato de tabela. É importante ter em mente, porém, que essas tabelas são internamente estruturadas como mostrado acima.

4.1.2 stops.txt

Arquivo usado para descrever as paradas de transporte público que compõem o sistema. Os pontos listados neste arquivo podem fazer menção a paradas mais simples (como pontos de ônibus), estações, plataformas, entradas e saídas de estações, etc. A Table 4.2 mostra o `stops.txt` do *feed* da SPTrans.

Table 4.2: Exemplo de arquivo `stops.txt`

stop_id	stop_name	stop_desc	stop_lat	stop_lon
706325	Parada 14 Bis B/C	Viad. Dr. Plínio De Queiroz, 901	- 23.55593	- 46.65011

stop_id	stop_name	stop_desc	stop_lat	stop_lon
810602	R. Sta. Rita, 56	Ref.: R. Bresser / R. João Boemer	-	-
			23.53337	46.61229
910776	Av. Do Estado, 5854	Ref.: Rua Dona Ana Néri	-	-
			23.55896	46.61520
1010092	Parada Caetano Pinto	Av. Rangel Pestana, 1249 Ref.: Rua Caetano Pinto/rua Prof. Batista De Andrade	-	-
			23.54615	46.62218
1010093	Parada Piratinunga	Av. Rangel Pestana, 1479 Ref.: Rua Monsenhor Andrade	-	-
			23.54509	46.62006
1010099	R. Xavantes, 612	Ref.: Rua Joli	-	-
			23.53545	46.61368

As colunas **stop_id** e **stop_name** servem como identificadores de cada parada, porém cumprem papéis distintos. O principal propósito da **stop_id** é identificar relações entre esta tabela e outras que compõem a especificação (como veremos mais à frente no arquivo **stop_times.txt**). Já a coluna **stop_name** serve como um identificador que seja facilmente reconhecido pelo usuário do sistema. Seus valores, portanto, costumam ser nomes de estações, nomes de pontos de interesse da cidade ou endereços (como no caso do *feed* da SPTrans).

A coluna **stop_desc**, presente no *feed* da SPTrans, é opcional e permite à agência de transporte adicionar alguma descrição de cada parada e do seu entorno para facilitar a sua identificação. As colunas **stop_lat** e **stop_lon**, por fim, são as responsáveis por associar cada parada a uma posição espacial, através de suas coordenadas geográficas de latitude e longitude.

Entre as colunas opcionais não presentes neste feed estão a **location_type** e a **parent_station**. A **location_type** é utilizada para denotar o tipo de localização a que cada ponto se refere. Quando ausente, todos os pontos são interpretados como paradas de transporte público, mas valores distintos podem ser usados para distinguir uma parada (**location_type** = 0) de uma estação (**location_type** = 1) ou uma área de embarque (**location_type** = 2), por exemplo. A coluna **parent_station** é utilizada para descrever relações de hierarquia entre dois pontos. Por exemplo, uma área de desembarque deve dizer a qual parada/plataforma ela pertence e uma parada/plataforma pode também, opcionalmente, listar a qual estação ela pertence.

4.1.3 routes.txt

Arquivo usado para descrever as linhas de transporte público que rodam no sistema descrito pelo arquivo GTFS, incluindo os modos de transporte utilizados em cada uma. A Table 4.3 mostra o **routes.txt** do *feed* da SPTrans.

Table 4.3: Exemplo de arquivo `routes.txt`

route_id	agency_id	route_short_name	route_long_name	route_type
CPTM L07	1	CPTM L07	JUNDIAI - LUZ	2
CPTM L08	1	CPTM L08	AMADOR BUENO - JULIO PRESTES	2
CPTM L09	1	CPTM L09	GRAJAU - OSASCO	2
CPTM L10	1	CPTM L10	RIO GRANDE DA SERRA - BRÁS	2
CPTM L11	1	CPTM L11	ESTUDANTES - LUZ	2
CPTM L12	1	CPTM L12	CALMON VIANA - BRAS	2

Assim como no caso do arquivo `stops.txt`, a tabela do `routes.txt` também possui diferentes colunas que apontam o identificador de cada linha (`route_id`) e o seu nome. Neste caso, no entanto, existem duas colunas de nome, a `route_short_name` e a `route_long_name`. A primeira diz respeito ao nome da linha, usualmente utilizado por passageiros no dia-a-dia, enquanto o segundo tende a ser um nome mais descritivo. A SPTrans, por exemplo, optou por destacar os pontos finais de cada linha nesta coluna. Podemos notar também que os mesmos valores se repetem nas colunas `route_id` e `route_short_name`, o que não é obrigatório e nem proibido - neste caso, o produtor do *feed* julgou que os nomes das linhas poderiam funcionar satisfatoriamente como identificadores por serem razoavelmente curtos e não se repetirem.

A coluna `agency_id` é a chave que permite relacionar a tabela das rotas com a tabela descrita no `agency.txt`. Ela faz menção a uma agência descrita naquele arquivo, no caso a agência de id 1 (a própria SPTrans). Esta coluna é opcional no caso de *feeds* em que existe apenas uma agência, porém é obrigatória no caso em que existem mais de uma. Imaginemos, por exemplo, um feed que descreve um sistema multimodal que conta com um corredor de metrô e diversas linhas de ônibus: uma configuração possível de `routes.txt` descreveria as linhas de metrô como de responsabilidade da operadora do metrô, e as de ônibus como de responsabilidade da empresa responsável pelo planejamento das linhas de ônibus, por exemplo.

A coluna `route_type` é utilizada para descrever o modo de transporte utilizado em cada linha. Esta coluna aceita diferentes números, cada um representando um determinado modo. Este exemplo descreve linhas de trem, cujo valor numérico correspondente é 2. Os valores válidos para esta coluna são listados na [especificação](#).

4.1.4 trips.txt

Arquivo usado para descrever as viagens realizadas pelo sistema de transporte público descrito pelo *feed*. A viagem é a unidade básica de movimento do formato GTFS: ela associa diferentes partidas de cada linha de transporte público a um serviço que opera em determinados dias da semana (como veremos mais à frente no arquivo `calendar.txt`) e uma trajetória de viagem (como veremos mais à frente no arquivo `shapes.txt`). A Table 4.4 mostra o `trips.txt` do *feed* da SPTrans.

Table 4.4: Exemplo de arquivo `trips.txt`

trip_id	route_id	service_id	trip_headsign	direction_id	shape_id
CPTM L07-0	CPTM L07	USD	JUNDIAI	0	17846
CPTM L07-1	CPTM L07	USD	LUZ	1	17847
CPTM L08-0	CPTM L08	USD	AMADOR BUENO	0	17848
CPTM L08-1	CPTM L08	USD	JULIO PRESTES	1	17849
CPTM L09-0	CPTM L09	USD	GRAJAU	0	17850
CPTM L09-1	CPTM L09	USD	OSASCO	1	17851

A coluna `trip_id` identifica cada uma das viagens descritas na tabela. A `route_id` faz referência a uma linha de transporte público identificada no arquivo `routes.txt`. A coluna `service_id` identifica serviços que determinam os dias da semana em que cada uma das viagens opera (dias úteis, finais de semana, uma mistura dos dois, etc.), descritos detalhadamente no arquivo `calendar.txt`. A última coluna à direita na tabela acima é a `shape_id`, que identifica a trajetória espacial de cada uma das viagens, descrita em detalhes no arquivo `shapes.txt`.

As duas colunas restantes, `trip_headsign` e `direction_id`, são opcionais e devem ser utilizadas para descrever o sentido/destino da viagem. A primeira, `trip_headsign`, é utilizada para ditar o texto que aparece no letreiro de veículos (no caso de um ônibus, por exemplo) ou em painéis informativos (como em metrô e trens) destacando o destino da viagem. Já a coluna `direction_id` é frequentemente utilizada em conjunto com a primeira para dar uma conotação de ida ou volta para cada viagem, onde 0 representa ida e 1 volta, ou vice-versa (assim como ida e volta são conceitos que mudam conforme o referencial, os valores 0 e 1 podem ser usados como desejado, desde que um represente um sentido e o outro o contrário). No caso do nosso exemplo, as duas primeiras linhas são viagens que fazem menção à mesma

rota de transporte público (CPTM L07), porém em sentidos opostos: uma corre em direção a Jundiaí, e a outra à Luz.

4.1.5 calendar.txt

Arquivo usado para descrever os diferentes tipos de serviço existentes no sistema de transporte público descrito pelo *feed*. Um serviço, neste contexto, denota um conjunto de dias da semana em que viagens são realizadas. Cada serviço também é definido pela data em que começa a valer e pela data a partir do qual ele não é mais válido. A Table 4.5 mostra o `calendar.txt` do *feed* da SPTrans.

Table 4.5: Exemplo de arquivo `calendar.txt`

service_id	monday	tuesday	wednesday	thursday	friday	saturday	sunday	start_date	end_date
USD	1	1	1	1	1	1	1	20080101	20200501
U__	1	1	1	1	1	0	0	20080101	20200501
US_	1	1	1	1	1	1	0	20080101	20200501
_SD	0	0	0	0	0	1	1	20080101	20200501
__D	0	0	0	0	0	0	1	20080101	20200501
S	0	0	0	0	0	1	0	20080101	20200501

A coluna `service_id` identifica cada um dos serviços descritos na tabela. Como mostrado anteriormente, este identificador é usado também no arquivo `trips.txt`, e é o responsável por associar cada viagem a um determinado serviço.

As colunas `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday` e `sunday` (segunda-feira a domingo, em inglês) são utilizadas para delimitar os dias em que cada serviço funciona. Um valor de 1 significa que o serviço opera em um determinado dia, enquanto um valor de 0 significa que ele não opera. Como podemos ver no exemplo acima, o serviço USD opera em todos os dias da semana. Já o serviço U__ opera apenas em dias úteis.

Por fim, as colunas `start_date` e `end_date` delimitam o intervalo em que cada serviço é válido. As datas do formato GTFS são sempre formatadas segundo a regra YYYYMMDD - ou seja, os primeiros quatro números definem o ano, os dois subsequentes definem o mês e os últimos dois, o dia. O valor 20220428, por exemplo, representa o dia 28 de abril de 2022.

4.1.6 shapes.txt

Arquivo usado para descrever a trajetória espacial de cada viagem listada no *feed*. Este arquivo é opcional, mas fortemente recomendado que agências de transporte o incluam em seus arquivos de GTFS. A Table 4.6 mostra o `shapes.txt` do *feed* da SPTrans.

Table 4.6: Exemplo de arquivo `shapes.txt`

shape_id	shape_pt_lat	shape_pt_lon	shape_pt_sequence
17846	-23.53517	-46.63535	1
17846	-23.53513	-46.63548	2
17846	-23.53494	-46.63626	3
17846	-23.53473	-46.63710	4
17846	-23.53466	-46.63735	5
17846	-23.53416	-46.63866	6

A coluna `shape_id` identifica cada uma das trajetórias descritas na tabela. Como mostrado anteriormente, este identificador é usado também no arquivo `trips.txt`, e é o responsável por associar cada viagem à sua trajetória espacial. Diferentemente de todos os outros identificadores que vimos até então, no entanto, o identificador `shape_id` se repete em diversas observações da tabela. Isso porque o arquivo apresenta para cada `shape_id` uma série de pontos espaciais, cujas coordenadas geográficas são apresentadas nas colunas `shape_pt_lat` e `shape_pt_lon`. Por sua vez, a coluna `shape_pt_sequence` lista a sequência na qual cada ponto se conecta para formar a trajetória de cada `shape_id`. Os valores listados nesta coluna devem ser ordenados de forma crescente.

4.1.7 stop_times.txt

Arquivo usado para descrever o cronograma com tabela de horários de cada viagem, incluindo o horário de chegada e partida em cada uma das paradas. A formatação deste arquivo depende da existência ou não de um arquivo `frequencies.txt`, o qual cobriremos a seguir. Por enquanto olharemos para o `stop_times.txt` do *feed* da SPTrans, que também conta com um `frequencies.txt`, na Table 4.7.

Table 4.7: Exemplo de arquivo `stop_times.txt`

trip_id	arrival_time	departure_time	stop_id	stop_sequence
CPTM L07-0	04:00:00	04:00:00	18940	1
CPTM L07-0	04:08:00	04:08:00	18920	2
CPTM L07-0	04:16:00	04:16:00	18919	3
CPTM L07-0	04:24:00	04:24:00	18917	4
CPTM L07-0	04:32:00	04:32:00	18916	5
CPTM L07-0	04:40:00	04:40:00	18965	6

A viagem cujo cronograma está sendo descrito é identificada pela coluna `trip_id`. De forma análoga ao que acontece na tabela de trajetórias, um mesmo `trip_id` se repete em muitas

observações da tabela porque, assim como uma trajetória é composta por uma sequência de pontos espaciais, um cronograma é composto por uma sequência de diversos horários de partida/chegada e paradas de transporte público.

As colunas seguintes, `arrival_time`, `departure_time` e `stop_id`, são as responsáveis por descrever o cronograma em si, associando um horário de chegada e um horário de partida para cada uma das paradas visitadas na viagem. As colunas de horário são formatadas segundo a regra `HH:MM:SS` - ou seja, os dois primeiros números definem a hora, os dois subsequentes os minutos e os últimos dois, os segundos. É importante ainda comentar que esta formatação aceita valores de hora maiores do que 24. Isto quer dizer que uma viagem cuja última parada seja visitada às 1h da manhã do dia seguinte ao que começou a rodar (digamos que ela tenha partido do ponto inicial às 23h) deve ter como horário de chegada `25:00:00`, e não `01:00:00`. A coluna `stop_id`, por sua vez, associa a parada visitada à uma descrição feita no arquivo `stops.txt`. Por fim, a coluna `stop_sequence` lista a sequência na qual cada parada se conecta às demais para formar o cronograma da viagem. Seus valores devem ser sempre ordenados de forma crescente.

Vale destacar aqui a diferença entre os arquivos `shapes.txt` e `stop_times.txt`. Embora os dois descrevam uma viagem espacialmente, eles o fazem de forma diferente. O `stop_times.txt` descreve a sequência de paradas e horários que compõem um cronograma, mas nada diz sobre o trajeto percorrido pelo veículo entre cada uma das paradas. Já o `shapes.txt` traz a trajetória da viagem como um todo, mas não descreve em que ponto do espaço estão as paradas da viagem. Quando usamos os dois arquivos em conjunto, portanto, sabemos não apenas o cronograma de cada viagem, mas também a trajetória da viagem entre paradas.

4.1.8 frequencies.txt

Arquivo opcional usado para descrever a frequência de cada viagem dentro de um determinado período do dia. A Table 4.8 mostra o `frequencies.txt` do *feed* da SPTrans.

Table 4.8: Exemplo de arquivo `frequencies.txt`

<code>trip_id</code>	<code>start_time</code>	<code>end_time</code>	<code>headway_secs</code>
CPTM L07-0	04:00:00	04:59:00	720
CPTM L07-0	05:00:00	05:59:00	360
CPTM L07-0	06:00:00	06:59:00	360
CPTM L07-0	07:00:00	07:59:00	360
CPTM L07-0	08:00:00	08:59:00	360
CPTM L07-0	09:00:00	09:59:00	480

A viagem cuja frequência está sendo descrita é identificada pela coluna `trip_id`. Novamente, um mesmo id pode aparecer em várias observações da tabela. Isso porque a especificação

GTFS prevê que uma mesma viagem pode ter frequências diferentes ao longo do dia (como em horários de pico e fora-pico, por exemplo). Assim, cada linha da tabela descreve um período do dia que se inicia no horário descrito na coluna `start_time` e termina no horário assinalado na `end_time`.

Dentro do período especificado por essas duas colunas a viagem possui um *headway* detalhado na coluna `headway_secs`. O *headway* é o tempo que separa a passagem de dois veículos que operam a mesma linha de transporte público. No caso desta tabela, esse tempo deve ser especificado em segundos. Um valor de 720 entre 04:00h e 05:00h, portanto, significa que dentro deste período a viagem CPTM L07-0 ocorre de 12 em 12 minutos.

É importante entender, agora, como a presença da tabela `frequencies.txt` altera a especificação da tabela `stop_times.txt`. Como podemos ver no nosso exemplo, o horário de partida definido no `stop_times.txt` para a viagem CPTM L07-0 é 04:00h, e ela chega na segunda parada às 04:08h. O cronograma de chegada e saída de uma mesma parada de uma viagem, no entanto, não pode ser definido mais de uma vez na tabela. Como então definir o cronograma das viagens que partem às 04:12h, 04:24h, 04:36h, etc. (lembrem-se que o *headway* desta viagem é de 12 minutos)?

No caso em que a frequência de uma viagem é especificada no `frequencies.txt`, o cronograma (a tabela de horários) de uma viagem definido no `stop_times.txt` deve ser entendido como uma referência que descreve o tempo entre paradas. Isto é, os horários ali definidos não devem ser interpretados à risca. Por exemplo, o cronograma listado estabelece que o tempo de viagem entre a primeira e a segunda parada é de 8 minutos, e o tempo entre a segunda e a terceira também. Ou seja, a viagem que parte da primeira parada às 04:00h chega na segunda às 04:08h, e na terceira às 04:16h. A próxima viagem, que parte da primeira parada às 04:12h, por sua vez, chega na segunda parada às 04:20h, e na terceira às 04:28h.

Assumindo partidas de 12 em 12 minutos a partir das 04:00h, por exemplo, poderíamos descrever as mesmas viagens no `stop_times.txt` sem fazer uso do arquivo `frequencies.txt`. Para isso poderíamos adicionar um sufixo que identificasse cada uma das viagens referentes à linha (`route_id`) com identificador CPTM L07 ao longo do dia. A viagem (`trip_id`) com identificador CPTM L07-0_1, por exemplo, seria a primeira viagem do dia e partiria da primeira parada às 04:00h e chegaria na segunda parada às 04:08h. A viagem CPTM L07-0_2, por sua vez, seria a segunda viagem e partiria da primeira parada às 04:12h e chegaria na segunda às 04:20h, e daí em diante. Cada uma dessas viagens deveria ser também adicionada ao arquivo `trips.txt` e a quaisquer outros que possuam a coluna `trip_id` como identificador.

Outra variável que afeta a forma na qual o `frequencies.txt` afeta as tabelas de horários na tabela `stop_times.txt` é a coluna opcional `exact_times`. Um valor de 0 nesta coluna (ou quando ela está ausente do *feed*, como no caso do arquivo GTFS da SPTrans) indica que a viagem não necessariamente segue um cronograma fixo ao longo do período. Ao invés disso, operadores tentam se ater a um determinado *headway* durante o período. Naquele exemplo de uma viagem cujo *headway* é de 12 minutos entre as 04:00h e 05:00h, isto significaria que não necessariamente a primeira partida sairá exatamente às 04:00h, a segunda às 04:12h, e por aí

em diante. A primeira pode, por exemplo, sair às 04:02h. A segunda, às 04:14h ou 04:13h, etc. Caso desejemos definir um cronograma que é seguido à risca, obtendo o mesmo resultado de definir diversas viagens semelhantes partindo em diferentes horários no `stop_times.txt` (como mostrado anteriormente), devemos utilizar o valor 1 na coluna `exact_times`.

4.2 Onde encontrar GTFS de cidades brasileiras

No Brasil, existem dados de GTFS para diversas cidades. Em muitos casos, no entanto, esses dados são de propriedade de empresas operadoras e concessionárias de transporte, e não do poder público. Infelizmente, esses arquivos raramente são disponibilizados abertamente e publicamente, contrariando boas práticas de gestão e compartilhamento de dados de interesse público. A Table 4.9 mostra as fontes dos dados GTFS de algumas das poucas cidades do Brasil que disponibilizavam seus feeds abertamente no momento desta publicação.

Table 4.9: Fontes de dados GTFS publicamente disponíveis no Brasil

Cidade	Fonte	Informações
Belo Horizonte	BHTrans	Dado aberto: transporte convencional ; transporte suplementar .
Fortaleza	ETUFOR	Dado aberto .
Fortaleza	Metrofor	Dado aberto .
Porto Alegre	EPTC	Dado aberto .
Rio de Janeiro	SMTR	Dado aberto .
São Paulo	EMTU	Download neste link . Necessário cadastro.
São Paulo	SPTrans	Download neste link . Necessário cadastro.

Obs. Os dados de GTFS disponibilizados para Rio de Janeiro e Porto Alegre não necessariamente cobrem todos os modos de transporte público disponíveis nessas cidades.

5 Manipulação e visualização de dados GTFS

Usualmente, arquivos GTFS oriundos de bases oficiais são utilizados para desenvolver análises e pesquisas que possuem diversos elementos comuns. Visando facilitar a leitura, o processamento e a análise desses dados, a equipe do Projeto Acesso a Oportunidades vêm desenvolvendo o pacote de R `gtfstools`¹, que oferece diversas funções que facilitam a manipulação e a exploração de feeds.

Neste capítulo, nós iremos passar por algumas das funcionalidades mais frequentemente utilizadas do pacote. Para isso, vamos utilizar uma amostra do *feed* da SPTrans apresentado no capítulo anterior, disponível dentro do `gtfstools`.

5.1 Leitura e manipulação básica de arquivos GTFS

A leitura de arquivos GTFS com o `gtfstools` é feita com a função `read_gtfs()`, que recebe uma *string* com o caminho do arquivo. Um *feed* é representado como uma lista de `data.tables`, uma versão de alta performance da classe `data.frame`. Ao longo deste capítulo, nós vamos nos referir a esta lista de tabelas como um **objeto GTFS**. Por padrão, a função lê todas as tabelas `.txt` do *feed*:

```
library(gtfstools)

path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")

gtfs <- read_gtfs(path)

names(gtfs)
```

```
[1] "agency"      "calendar"    "frequencies" "routes"      "shapes"
[6] "stop_times"  "stops"       "trips"
```

Como podemos ver, cada `data.table` dentro do objeto GTFS é nomeado de acordo com a tabela que ele representa, porém sem a extensão `.txt`. Isso nos permite selecionar e manipular

¹Disponível em <https://github.com/ipeaGIT/gtfstools>.

cada uma das tabelas separadamente. O código abaixo, por exemplo, mostra os 6 primeiros registros da tabela *trips*:

```
head(gtfs$trips)
```

	route_id	service_id	trip_id	trip_headsign	direction_id	shape_id
1:	CPTM L07	USD	CPTM L07-0	JUNDIAI	0	17846
2:	CPTM L07	USD	CPTM L07-1	LUZ	1	17847
3:	CPTM L08	USD	CPTM L08-0	AMADOR BUENO	0	17848
4:	CPTM L08	USD	CPTM L08-1	JULIO PRESTES	1	17849
5:	CPTM L09	USD	CPTM L09-0	GRAJAU	0	17850
6:	CPTM L09	USD	CPTM L09-1	OSASCO	1	17851

As tabelas dentro de um objeto GTFS podem ser facilmente manipuladas usando a sintaxe de tabelas `data.table`. O pacote `data.table` oferece diversas funcionalidades úteis, como a edição de colunas por referência, filtros de linhas muito rápidos e agregação de dados eficiente². Para adicionar 100 segundos a todos os *headways* listados na tabela *frequencies* e reverter essa mudança em seguida, por exemplo, nós podemos usar o código abaixo:

```
# original
original_headway <- gtfs$frequencies$headway_secs
head(gtfs$frequencies, 3)
```

	trip_id	start_time	end_time	headway_secs
1:	CPTM L07-0	04:00:00	04:59:00	720
2:	CPTM L07-0	05:00:00	05:59:00	360
3:	CPTM L07-0	06:00:00	06:59:00	360

```
# modified
gtfs$frequencies[, headway_secs := headway_secs + 100]
head(gtfs$frequencies, 3)
```

	trip_id	start_time	end_time	headway_secs
1:	CPTM L07-0	04:00:00	04:59:00	820
2:	CPTM L07-0	05:00:00	05:59:00	460
3:	CPTM L07-0	06:00:00	06:59:00	460

²Mais detalhes sobre o uso e a sintaxe do `data.table` podem ser lidos em <https://rdatatable.gitlab.io/data.table/index.html>.

```
# back to original
gtfs$frequencies[, headway_secs := original_headway]
head(gtfs$frequencies, 3)
```

```
      trip_id start_time end_time headway_secs
1: CPTM L07-0  04:00:00 04:59:00           720
2: CPTM L07-0  05:00:00 05:59:00           360
3: CPTM L07-0  06:00:00 06:59:00           360
```

Ao final de edições de um objeto GTFS no R, frequentemente vamos querer usar o GTFS manipulado para fazer análises de diferentes tipos. Para isso, é comum que precisemos do arquivo GTFS em formato `.zip` novamente, e não da lista de tabelas dentro do R. O pacote disponibiliza a função `write_gtfs()` exatamente com a finalidade de transformar objetos GTFS que existem apenas dentro do R em arquivos GTFS salvos na memória de seu computador. Para usá-la, é necessário apenas listar o objeto e o endereço no qual ele deve ser salvo:

```
dest_path <- tempfile("new_gtfs", fileext = ".zip")

file.exists(dest_path)
```

```
[1] FALSE
```

```
write_gtfs(gtfs, dest_path)

file.exists(dest_path)
```

```
[1] TRUE
```

```
zip::zip_list(dest_path)[, c("filename", "compressed_size", "timestamp")]
```

```
      filename compressed_size      timestamp
1   agency.txt           112 2022-09-29 20:34:16
2  calendar.txt           129 2022-09-29 20:34:16
3 frequencies.txt        2381 2022-09-29 20:34:16
4    routes.txt           659 2022-09-29 20:34:16
5   shapes.txt       160470 2022-09-29 20:34:16
6 stop_times.txt          7907 2022-09-29 20:34:16
7    stops.txt       18797 2022-09-29 20:34:16
8    trips.txt           717 2022-09-29 20:34:16
```

5.2 Cálculo de velocidade das linhas

Arquivos GTFS são frequentemente utilizados em estimativas de roteamento de transporte público e para informar passageiros sobre a tabela de horários das diferentes rotas que operam em uma região. Dessa forma, é extremamente importante que o cronograma das viagens e a velocidade operacional de cada linha estejam adequadamente descritos no *feed*.

O `gtfstools` disponibiliza a função `get_trip_speed()` para facilitar o cálculo da velocidade de cada viagem presente no *feed*. Por padrão a função retorna a velocidade (em km/h) de todas as viagens do GTFS, mas viagens individuais também podem ser especificadas:

```
speeds <- get_trip_speed(gtfs)
```

```
head(speeds)
```

	trip_id	origin_file	speed
1:	2002-10-0	shapes	8.952511
2:	2105-10-0	shapes	10.253365
3:	2105-10-1	shapes	9.795292
4:	2161-10-0	shapes	11.182534
5:	2161-10-1	shapes	11.784458
6:	4491-10-0	shapes	13.203560

```
nrow(speeds)
```

```
[1] 36
```

```
speeds <- get_trip_speed(gtfs, trip_id = c("CPTM L07-0", "2002-10-0"))
```

```
speeds
```

	trip_id	origin_file	speed
1:	2002-10-0	shapes	8.952511
2:	CPTM L07-0	shapes	26.787768

Calcular a velocidade de uma viagem requer que nós saibamos o seu comprimento e em quanto tempo ela foi realizada. Para isso, portanto, a `get_trip_speed()` utiliza duas outras funções

do `gtfstools` por trás dos panos: a `get_trip_length()` e a `get_trip_duration()`. O funcionamento das duas é muito parecido com o mostrado anteriormente, retornando o comprimento/duração de todas as viagens por padrão, ou de apenas algumas selecionadas, caso desejado. Abaixo nós mostramos seus comportamentos padrões:

```
length <- get_trip_length(gtfs, file = "shapes")  
  
head(length)
```

	trip_id	length	origin_file
1:	CPTM L07-0	60.71894	shapes
2:	CPTM L07-1	60.71894	shapes
3:	CPTM L08-0	41.79037	shapes
4:	CPTM L08-1	41.79037	shapes
5:	CPTM L09-0	31.88906	shapes
6:	CPTM L09-1	31.88906	shapes

```
duration <- get_trip_duration(gtfs)  
  
head(duration)
```

	trip_id	duration
1:	2002-10-0	48
2:	2105-10-0	108
3:	2105-10-1	111
4:	2161-10-0	94
5:	2161-10-1	93
6:	4491-10-0	69

Assim como a `get_trip_speed()` retorna velocidades em km/h por padrão, a `get_trip_length()` retorna os comprimentos em km e a `get_trip_duration()` retorna a duração em minutos. Essas unidades podem ser ajustadas com o argumento `unit`, presente em todas as funções.

5.3 Combinação e filtragem de *feeds*

Muitas vezes o processo de processamento e edição de arquivos GTFS é realizado, em grande medida, manualmente. Consequentemente, pequenas inconsistências podem passar batidas pelos responsáveis por esse processamento. Um problema comumente observado em *feeds* é

a presença de registros duplicados em uma mesma tabela. O *feed* da SPTrans, por exemplo, possui registros duplicados tanto no *agency.txt* quanto no *calendar.txt*:

```
gtfs$agency
```

```

agency_id agency_name agency_url
1:         1    SPTRANS http://www.sptrans.com.br/?versao=011019
2:         1    SPTRANS http://www.sptrans.com.br/?versao=011019
  agency_timezone agency_lang
1: America/Sao_Paulo      pt
2: America/Sao_Paulo      pt

```

```
gtfs$calendar
```

```

service_id monday tuesday wednesday thursday friday saturday sunday
1:         USD      1        1          1          1          1          1
2:         U__      1        1          1          1          1          0
3:         US_      1        1          1          1          1          0
4:         _SD      0        0          0          0          0          1
5:         __D      0        0          0          0          0          0
6:         _S_      0        0          0          0          0          1
7:         USD      1        1          1          1          1          1
8:         U__      1        1          1          1          1          0
9:         US_      1        1          1          1          1          0
10:        _SD      0        0          0          0          0          1
11:        __D      0        0          0          0          0          0
12:        _S_      0        0          0          0          0          1

start_date end_date
1: 2008-01-01 2020-05-01
2: 2008-01-01 2020-05-01
3: 2008-01-01 2020-05-01
4: 2008-01-01 2020-05-01
5: 2008-01-01 2020-05-01
6: 2008-01-01 2020-05-01
7: 2008-01-01 2020-05-01
8: 2008-01-01 2020-05-01
9: 2008-01-01 2020-05-01
10: 2008-01-01 2020-05-01
11: 2008-01-01 2020-05-01
12: 2008-01-01 2020-05-01

```

O `gtfstools` disponibiliza a função `remove_duplicates()` para remover essas duplicatas. Esta função recebe como *input* um objeto GTFS e retorna o mesmo objeto, porém sem registros duplicados:

```
no_dups_gtfs <- remove_duplicates(gtfs)

no_dups_gtfs$agency

  agency_id agency_name          agency_url
1:         1   SPTRANS http://www.sptrans.com.br/?versao=011019
  agency_timezone agency_lang
1: America/Sao_Paulo      pt

no_dups_gtfs$calendar

  service_id monday tuesday wednesday thursday friday saturday sunday
1:      USD      1      1      1      1      1      1      1
2:      U__      1      1      1      1      1      0      0
3:      US_      1      1      1      1      1      1      0
4:      _SD      0      0      0      0      0      1      1
5:      __D      0      0      0      0      0      0      1
6:      _S_      0      0      0      0      0      1      0
  start_date  end_date
1: 2008-01-01 2020-05-01
2: 2008-01-01 2020-05-01
3: 2008-01-01 2020-05-01
4: 2008-01-01 2020-05-01
5: 2008-01-01 2020-05-01
6: 2008-01-01 2020-05-01
```

Frequentemente, também, lidamos com múltiplos *feeds* em uma mesma área de estudo. Neste caso, muitas vezes gostaríamos de uni-los em um único arquivo, diminuindo assim o esforço de manipulação e processamento dos dados. Para isso, o `gtfstools` disponibiliza a função `merge_gtfs()`. O exemplo abaixo mostra o resultado da combinação de dois *feeds* distintos, o da SPTrans (sem duplicatas) e o da EMTU, de Porto Alegre:

```
poa_path <- system.file("extdata/poa_gtfs.zip", package = "gtfstools")
poa_gtfs <- read_gtfs(poa_path)

poa_gtfs$agency
```

```

agency_id          agency_name          agency_url
1:      EPTC Empresa Publica de Transportes e Circulação http://www.eptc.com.br
agency_timezone agency_lang agency_phone
1: America/Sao_Paulo      pt      156
agency_fare_url
1: http://www2.portoalegre.rs.gov.br/eptc/default.php?p_secao=155

```

```
no_dups_gtfs$agency
```

```

agency_id agency_name          agency_url
1:      1      SPTRANS http://www.sptrans.com.br/?versao=011019
agency_timezone agency_lang
1: America/Sao_Paulo      pt

```

```
combined_gtfs <- merge_gtfs(no_dups_gtfs, poa_gtfs)
```

```
combined_gtfs$agency
```

```

agency_id          agency_name
1:      1      SPTRANS
2:      EPTC Empresa Publica de Transportes e Circulação
agency_url  agency_timezone agency_lang
1: http://www.sptrans.com.br/?versao=011019 America/Sao_Paulo      pt
2:      http://www.eptc.com.br America/Sao_Paulo      pt
agency_phone          agency_fare_url
1:
2:      156 http://www2.portoalegre.rs.gov.br/eptc/default.php?p_secao=155

```

Como podemos ver, os registros das tabelas de ambos os *feeds* foram combinados em uma única tabela. Este é o caso quando os dois (ou mais, caso desejado) objetos GTFS possuem registros de uma mesma tabela (a *agency*, no exemplo). Caso apenas um dos objetos possua uma das tabelas, o resultado da operação de combinação copia esta tabela para o resultado final. É o caso, por exemplo, da tabela *frequencies*, que existe apenas no *feed* da SPTrans, mas não no da EMTU:

```
names(poa_gtfs)
```

```

[1] "agency"      "calendar"    "routes"      "shapes"      "stop_times"
[6] "stops"      "trips"

```

```
names(no_dups_gtfs)
```

```
[1] "agency"      "calendar"    "frequencies" "routes"      "shapes"
[6] "stop_times"  "stops"       "trips"
```

```
names(combined_gtfs)
```

```
[1] "agency"      "calendar"    "frequencies" "routes"      "shapes"
[6] "stop_times"  "stops"       "trips"
```

```
identical(no_dups_gtfs$frequencies, combined_gtfs$frequencies)
```

```
[1] TRUE
```

Um outro tipo de operação muito utilizada no tratamento de arquivos GTFS é o de filtragem desses arquivos. Frequentemente, *feeds* são usados para descrever redes de transporte público de grandíssima escala, transformando sua edição, análise e transferência em operações complexas. Por esse motivo, pesquisadores e planejadores muitas vezes precisam trabalhar com um subconjunto de dados descritos nos *feeds*. Caso desejemos estimar a acessibilidade de uma determinada região no horário de pico da manhã, por exemplo, podemos filtrar o nosso arquivo GTFS de modo a manter apenas os registros referentes a viagens que ocorrem nesse intervalo do dia.

O pacote `gtfstools` também traz diversas funções para facilitar a filtragem de arquivos GTFS. São elas:

- `filter_by_agency_id()`
- `filter_by_route_id()`
- `filter_by_service_id()`
- `filter_by_shape_id()`
- `filter_by_stop_id()`
- `filter_by_trip_id()`
- `filter_by_route_type()`
- `filter_by_weekday()`
- `filter_by_time_of_day()`
- `filter_by_sf()`

As seis primeiras (`filter_by_agency_id()`, `filter_by_route_id()`, `filter_by_service_id()`, `filter_by_shape_id()`, `filter_by_stop_id()` e `filter_by_trip_id()`) funcionam de forma muito similar. O usuário deve especificar uma vetor de identificadores, e a função mantém no objeto GTFS apenas os registros referentes a esses identificadores. O exemplo abaixo demonstra essa funcionalidade com a `filter_by_trip_id()`:

```
lobstr::obj_size(gtfs)
```

770.22 kB

```
head(gtfs$trips[, .(trip_id, trip_headsign, shape_id)])
```

	trip_id	trip_headsign	shape_id
1:	CPTM L07-0	JUNDIAI	17846
2:	CPTM L07-1	LUZ	17847
3:	CPTM L08-0	AMADOR BUENO	17848
4:	CPTM L08-1	JULIO PRESTES	17849
5:	CPTM L09-0	GRAJAU	17850
6:	CPTM L09-1	OSASCO	17851

```
smaller_gtfs <- filter_by_trip_id(gtfs, trip_id = c("CPTM L07-0", "CPTM L07-1"))
```

```
lobstr::obj_size(smaller_gtfs)
```

61.62 kB

```
head(smaller_gtfs$trips[, .(trip_id, trip_headsign, shape_id)])
```

	trip_id	trip_headsign	shape_id
1:	CPTM L07-0	JUNDIAI	17846
2:	CPTM L07-1	LUZ	17847

```
unique(smaller_gtfs$shapes$shape_id)
```

```
[1] "17846" "17847"
```

O código acima mostra que a função não filtra apenas a tabela `trips`, mas também as outras tabelas que fazem referência aos identificadores especificados. Por exemplo, a trajetória das viagens CPTM L07-0 and CPTM L07-1 é descrita pelos `shape_ids` 17846 and 17847, respectivamente. Esses são, portanto, os únicos identificadores da tabela `shapes` mantidos no GTFS filtrado.

A função também funciona com o comportamento diametralmente oposto: em vez de definirmos os identificadores cujos registros devem ser *mantidos* no *feed*, especificamos os identificadores que devem ser *retirados* dele. Para isso, usamos o argumento `keep` com valor `FALSE`:

```
smaller_gtfs <- filter_by_trip_id(
  gtfs,
  c("CPTM L07-0", "CPTM L07-1"),
  keep = FALSE
)

head(smaller_gtfs$trips[, .(trip_id, trip_headsign, shape_id)])
```

	trip_id	trip_headsign	shape_id
1:	CPTM L08-0	AMADOR BUENO	17848
2:	CPTM L08-1	JULIO PRESTES	17849
3:	CPTM L09-0	GRAJAU	17850
4:	CPTM L09-1	OSASCO	17851
5:	CPTM L10-0	RIO GRANDE DA SERRA	17852
6:	CPTM L10-1	BRÁS	17853

```
head(unique(smaller_gtfs$shapes$shape_id))
```

```
[1] "17848" "17849" "17850" "17851" "17852" "17853"
```

Como podemos ver, as viagens especificadas, bem como suas trajetórias, não estão presentes no GTFS filtrado. A mesma lógica aqui demonstrada com a `filter_by_trip_id()` é válida para as funções que filtram objetos GTFS pelos identificadores `agency_id`, `route_id`, `service_id`, `shape_id`, `stop_id` e `route_type`.

Outra operação que recorrentemente aparece em análises que envolvem dados GTFS é a de manter serviços que funcionem apenas em determinados horários do dia ou dias da semana. Para isso, o pacote disponibiliza as funções `filter_by_weekday()` e `filter_by_time_of_day()`.

A `filter_by_weekday()` recebe os dias da semana (em inglês) cujos serviços que neles operam devem ser mantidos. Adicionalmente, a função também inclui o argumento `combine`, que define como filtros de dois ou mais dias funcionam. Quando este recebe o valor `"and"`, apenas serviços

que operam em todos os dias especificados são mantidos. Quando recebe o valor "or", serviços que operam em pelo menos um dos dias são mantidos:

```
smaller_gtfs <- filter_by_weekday(  
  no_dups_gtfs,  
  weekday = c("saturday", "sunday"),  
  combine = "and"  
)  
  
smaller_gtfs$calendar[, c("service_id", "sunday", "saturday")]
```

	service_id	sunday	saturday
1:	USD	1	1
2:	_SD	1	1

```
smaller_gtfs <- filter_by_weekday(  
  no_dups_gtfs,  
  weekday = c("sunday", "saturday"),  
  combine = "or"  
)  
  
smaller_gtfs$calendar[, c("service_id", "sunday", "saturday")]
```

	service_id	sunday	saturday
1:	USD	1	1
2:	US_	0	1
3:	_SD	1	1
4:	__D	1	0
5:	_S_	0	1

A `filter_by_time_of_day()`, por sua vez, recebe o começo e o final de uma janela de tempo e mantém os registros relacionados a viagens que rodam dentro dessa janela. O funcionamento da função depende da presença ou não da tabela `frequencies` no GTFS: o cronograma descrito na `stop_times` das viagens descritas na tabela `frequencies` não deve ser filtrado, pois, como comentado no capítulo anterior, ele serve como um modelo que dita o tempo de viagem entre uma parada e outra. Caso a `frequencies` esteja ausente, no entanto, a `stop_times` é filtrada segundo o intervalo de tempo especificado. Vamos ver como isso funciona com um exemplo:

```
smaller_gtfs <- filter_by_time_of_day(gtfs, from = "05:00:00", to = "06:00:00")
```

```
head(smaller_gtfs$frequencies)
```

	trip_id	start_time	end_time	headway_secs
1:	CPTM L07-0	05:00:00	05:59:00	360
2:	CPTM L07-1	05:00:00	05:59:00	360
3:	CPTM L08-0	05:00:00	05:59:00	480
4:	CPTM L08-1	05:00:00	05:59:00	480
5:	CPTM L09-0	05:00:00	05:59:00	480
6:	CPTM L09-1	05:00:00	05:59:00	480

```
head(smaller_gtfs$stop_times[, c("trip_id", "departure_time", "arrival_time")])
```

	trip_id	departure_time	arrival_time
1:	CPTM L07-0	04:00:00	04:00:00
2:	CPTM L07-0	04:08:00	04:08:00
3:	CPTM L07-0	04:16:00	04:16:00
4:	CPTM L07-0	04:24:00	04:24:00
5:	CPTM L07-0	04:32:00	04:32:00
6:	CPTM L07-0	04:40:00	04:40:00

```
frequencies <- gtfs$frequencies  
gtfs$frequencies <- NULL
```

```
smaller_gtfs <- filter_by_time_of_day(gtfs, from = "05:00:00", to = "06:00:00")
```

```
head(smaller_gtfs$stop_times[, c("trip_id", "departure_time", "arrival_time")])
```

	trip_id	departure_time	arrival_time
1:	CPTM L07-0	05:04:00	05:04:00
2:	CPTM L07-0	05:12:00	05:12:00
3:	CPTM L07-0	05:20:00	05:20:00
4:	CPTM L07-0	05:28:00	05:28:00
5:	CPTM L07-0	05:36:00	05:36:00
6:	CPTM L07-0	05:44:00	05:44:00

O filtro da tabela `stop_times` pode funcionar de duas formas distintas. Uma opção é manter intactas todas as viagens que cruzam a janela de tempo especificada. A outra é manter apenas os segmentos de viagens que ocorrem dentro da janela (comportamento padrão da função). Este comportamento é controlado com o parâmetro `full_trips`, como mostrado a seguir (prestem atenção nos horários e nos segmentos presentes em cada exemplo):

```

smaller_gtfs <- filter_by_time_of_day(
  gtfs,
  "05:00:00",
  "06:00:00",
  full_trips = TRUE
)

head(
  smaller_gtfs$stop_times[
    ,
    c("trip_id", "departure_time", "arrival_time", "stop_sequence")
  ]
)

```

	trip_id	departure_time	arrival_time	stop_sequence
1:	CPTM L07-0	04:00:00	04:00:00	1
2:	CPTM L07-0	04:08:00	04:08:00	2
3:	CPTM L07-0	04:16:00	04:16:00	3
4:	CPTM L07-0	04:24:00	04:24:00	4
5:	CPTM L07-0	04:32:00	04:32:00	5
6:	CPTM L07-0	04:40:00	04:40:00	6

```

smaller_gtfs <- filter_by_time_of_day(
  gtfs,
  "05:00:00",
  "06:00:00",
  full_trips = FALSE
)

head(
  smaller_gtfs$stop_times[
    ,
    c("trip_id", "departure_time", "arrival_time", "stop_sequence")
  ]
)

```

	trip_id	departure_time	arrival_time	stop_sequence
1:	CPTM L07-0	05:04:00	05:04:00	9
2:	CPTM L07-0	05:12:00	05:12:00	10
3:	CPTM L07-0	05:20:00	05:20:00	11

4: CPTM L07-0	05:28:00	05:28:00	12
5: CPTM L07-0	05:36:00	05:36:00	13
6: CPTM L07-0	05:44:00	05:44:00	14

Por fim, o pacote também disponibiliza uma função que permite filtrar o objeto GTFS usando um polígono espacial. A `filter_by_sf()` recebe um objeto do tipo `sf/sfc` (representação espacial estabelecida pelo pacote `sf`), ou sua *bounding box*, e mantém os registros cujas viagens são selecionadas por uma operação espacial especificada pelo usuário. Embora aparentemente complicado, este processo de filtragem é muito facilmente compreendido quando apresentado visualmente. Para isso, vamos criar uma função auxiliar:

```
library(ggplot2)

plotter <- function(gtfs,
                    geom,
                    spatial_operation = sf::st_intersects,
                    keep = TRUE,
                    do_filter = TRUE) {
  if (do_filter) {
    gtfs <- filter_by_sf(gtfs, geom, spatial_operation, keep)
  }

  shapes <- convert_shapes_to_sf(gtfs)
  trips <- get_trip_geometry(gtfs, file = "stop_times")
  geom <- sf::st_as_sfc(geom)

  ggplot() +
    geom_sf(data = trips) +
    geom_sf(data = shapes) +
    geom_sf(data = geom, fill = NA)
}
```

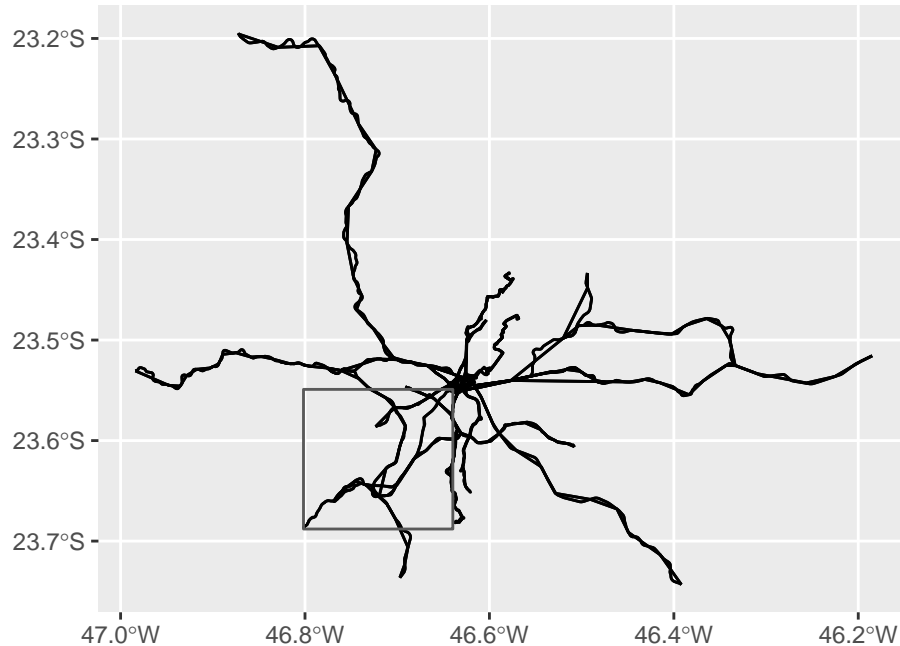
Esta função:

- Possui os mesmos parâmetros da `filter_by_sf()`, com a exceção do parâmetro `do_filter`, usado para mostrar os dados não filtrados;
- Condicionalmente filtra um objeto GTFS usando uma *bounding box* chamada `geom`;
- Gera a geometria das viagens usando as funções `convert_shapes_to_sf()` e `get_trip_geometry()`;
- Cria um polígono a partir da *bounding box*;

Vamos usar como exemplo a filtragem usando a *bounding box* da trajetória de *id* 68962. O código abaixo apresenta a distribuição espacial dos dados não filtrados:

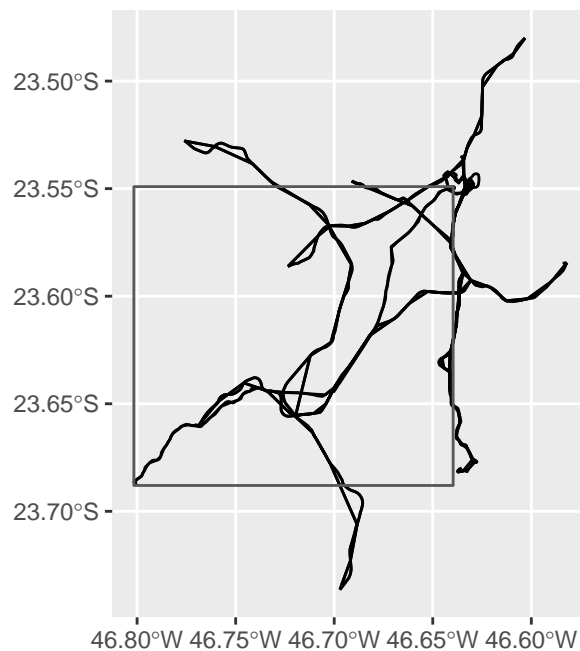
```
bbox <- sf::st_bbox(convert_shapes_to_sf(gtfs, shape_id = "68962"))

plotter(gtfs, bbox, do_filter = FALSE)
```



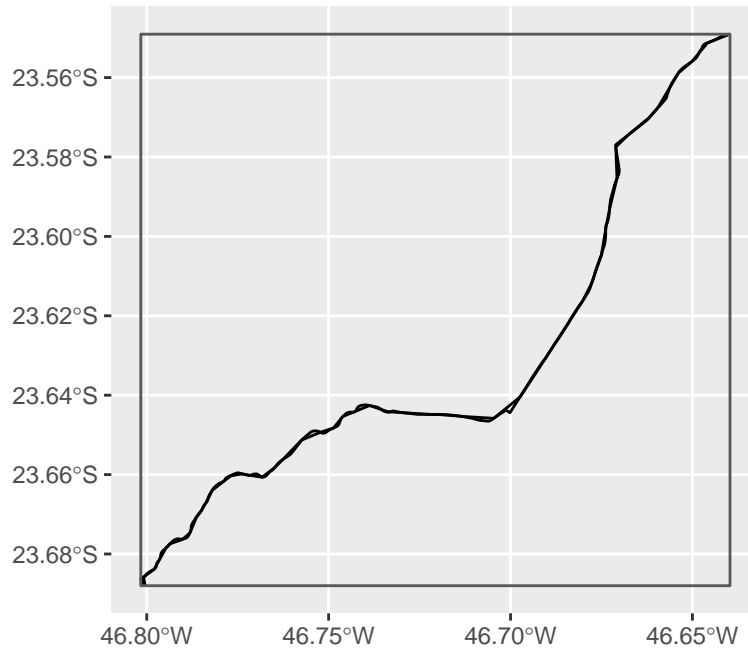
Notem aqui que nós usamos a função `convert_shapes_to_sf()`, que converte uma determinada trajetória descrita no GTFS em um objeto espacial do tipo `sf`, para obter a *bounding box* da trajetória. Por padrão, a `filter_by_sf()` (e a `plotter()`, consequentemente) mantém os dados relacionados aos registros de viagens cujas trajetórias possuem alguma interseção com o polígono espacial selecionado:

```
plotter(gtfs, bbox)
```



Nós podemos, no entanto, controlar a operação espacial usada no processo de filtragem. Por exemplo, o código abaixo mostra como nós podemos manter os dados relacionados a viagens que estão *contidas* dentro do polígono espacial:

```
plotter(gtfs, bbox, spatial_operation = sf::st_contains)
```

5.4 Mapeamento do *headway* das linhas

Como mostrado nas seções anteriores, o `gtfstools` disponibiliza uma grande caixa de ferramentas que podem ser usadas no processamento e na análise de arquivos GTFS. O pacote, no entanto, oferece diversas outras funções que não puderam ser apresentadas neste livro, por questões de espaço. A lista completa de funções disponíveis no pacote pode ser conferida no [site do gtfstools](#).

A apresentação das funções feitas até aqui tem um importante caráter demonstrativo, porém não mostra como elas podem ser usadas de forma conjunta no desenvolvimento de uma análise de um arquivo GTFS. Esta seção preenche esta lacuna, mostrando como o pacote pode ser usado em uma análise simples, que visa responder a seguinte pergunta: como se distribuem espacialmente os tempos entre veículos de uma mesma linha (ou seja, o *headway*) no GTFS da SPTrans?

A primeira etapa é definir o escopo da nossa análise. A fim de exemplo, vamos considerar o *headway* no pico da manhã, entre 7h e 9h, em uma típica terça-feira de operação. Para isso, precisamos filtrar o nosso *feed*:

```
gtfs <- read_gtfs(path)

filtered_gtfs <- gtfs |>
```

```

remove_duplicates() |>
filter_by_weekday("tuesday") |>
filter_by_time_of_day(from = "07:00:00", to = "09:00:00")

filtered_gtfs$frequencies[trip_id == "2105-10-0"]

```

	trip_id	start_time	end_time	headway_secs
1:	2105-10-0	07:00:00	07:59:00	900
2:	2105-10-0	08:00:00	08:59:00	1200

```
filtered_gtfs$calendar
```

	service_id	monday	tuesday	wednesday	thursday	friday	saturday	sunday
1:	USD	1	1	1	1	1	1	1
2:	U__	1	1	1	1	1	0	0

	start_date	end_date
1:	2008-01-01	2020-05-01
2:	2008-01-01	2020-05-01

Em seguida, precisamos calcular o *headway* dentro do período estabelecido. Essa informação pode ser encontrada na tabela *frequencies*, porém há um elemento complicador: cada viagem está associada a mais de um *headway*, como podemos ver acima. Para resolver esta questão, portanto, vamos calcular o *headway* médio* neste período.

Os primeiros registros da tabela *frequencies* do GTFS da SPTrans parecem sugerir que os períodos do dia estão listados sempre de uma em uma hora, porém isto não é uma regra estabelecida na especificação GTFS e nem é a prática adotada em outros GTFS. Por isso, nós vamos calcular a *média ponderada* do *headway* no período especificado. Para isso, nós precisamos multiplicar cada *headway* pelo intervalo de tempo em que ele é válido, e dividir o total desta soma pelo intervalo de tempo total (duas horas). Para calcular o intervalo de tempo em que cada *headway* é válido, nós usamos a função `convert_time_to_seconds()` para calcular o começo e o fim do intervalo em segundos e subtraímos o valor do fim pelo do começo, como abaixo:

```

filtered_gtfs <- convert_time_to_seconds(filtered_gtfs)

filtered_gtfs$frequencies[trip_id == "2105-10-0"]

```

	trip_id	start_time	end_time	headway_secs	start_time_secs	end_time_secs
1:	2105-10-0	07:00:00	07:59:00	900	25200	28740
2:	2105-10-0	08:00:00	08:59:00	1200	28800	32340

```
filtered_gtfs$frequencies[, time_interval := end_time_secs - start_time_secs]
```

Em seguida, nós calculamos o *headway* médio:

```
avg_headway <- filtered_gtfs$frequencies[
  ,
  .(avg_headway = weighted.mean(headway_secs, w = time_interval)),
  by = trip_id
]

avg_headway[trip_id == "2105-10-0"]
```

```
      trip_id avg_headway
1: 2105-10-0          1050
```

```
head(avg_headway)
```

```
      trip_id avg_headway
1: CPTM L07-0          360
2: CPTM L07-1          360
3: CPTM L08-0          300
4: CPTM L08-1          300
5: CPTM L09-0          240
6: CPTM L09-1          240
```

Nós precisamos agora gerar a trajetória espacial de cada viagem e juntar esta informação à do *headway* médio. Para isso, nós vamos utilizar a função `get_trip_geometry()`, que, dado um objeto GTFS, retorna a trajetória espacial de suas viagens. Esta função nos permite especificar de quais viagens nós queremos as trajetórias, logo nós vamos calcular apenas as daquelas que estão presentes na tabela de *headways* médios:

```
selected_trips <- avg_headway$trip_id

geoms <- get_trip_geometry(
  filtered_gtfs,
  trip_id = selected_trips,
  file = "shapes"
)

head(geoms)
```

Simple feature collection with 6 features and 2 fields

Geometry type: LINESTRING

Dimension: XY

Bounding box: xmin: -46.98404 ymin: -23.73644 xmax: -46.63535 ymax: -23.19474

Geodetic CRS: WGS 84

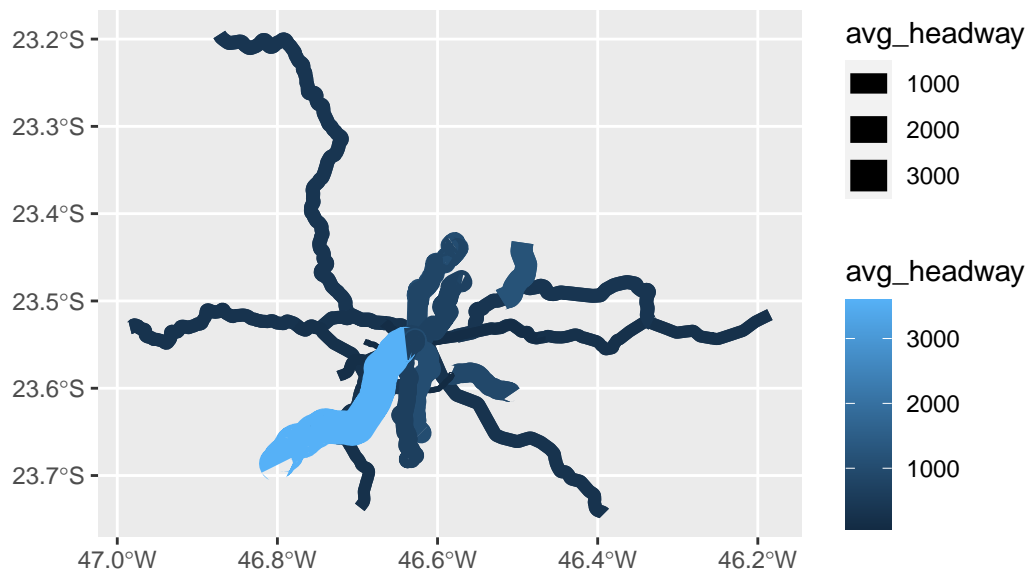
	trip_id	origin_file	geometry
1	CPTM L07-0	shapes	LINESTRING (-46.63535 -23.5...
2	CPTM L07-1	shapes	LINESTRING (-46.87255 -23.1...
3	CPTM L08-0	shapes	LINESTRING (-46.64073 -23.5...
4	CPTM L08-1	shapes	LINESTRING (-46.98404 -23.5...
5	CPTM L09-0	shapes	LINESTRING (-46.77604 -23.5...
6	CPTM L09-1	shapes	LINESTRING (-46.69711 -23.7...

O objeto `geoms` está no formato `sf`, e não no `data.table`, que precisamos que ele esteja para juntarmos à tabela de *headways*. Depois de convertê-lo para o formato desejado adequado e juntá-lo à tabela de *headways*, nós precisamos apenas configurar o nosso mapa como desejado. No exemplo abaixo, nós usamos cores e espessuras de linhas que variam de acordo com o *headway* de cada viagem:

```
library(data.table)
library(sf)

setDT(geoms)
avg_headway[geoms, on = "trip_id", geometry := i.geometry]

ggplot(st_sf(avg_headway)) +
  geom_sf(aes(color = avg_headway, size = avg_headway))
```



Como podemos ver, o pacote `gtfstools` torna o desenvolvimento de análises de *feeds* de transporte público algo fácil e que requer apenas o conhecimento básico de pacotes de manipulação de tabela (como o `data.table`) para grande parte das etapas que as compõem. O exemplo apresentado nesta seção mostra como muitas de suas funções podem ser usadas conjuntamente para revelar aspectos importantes de sistemas de transporte público descritos no formato GTFS.

Part IV

SEÇÃO 4: Avaliação de impacto

Objetivo: O objetivo desta seção é mostrar como avaliar o impacto de políticas urbanas sobre acessibilidade urbana usando R.

Embora muito utilizado na literatura científica, o conceito de acessibilidade ainda não é tão utilizado no planejamento e operação de sistemas de transporte público por parte de agências de transportes e tomadores de decisão. Muito disto se dá pela dificuldade de incorporar análises de acessibilidade a métodos de avaliação de projetos comumente utilizados, como análises de custo e benefício, por exemplo. Esta seção apresenta um método de avaliação de impactos de projetos e investimentos de transporte que tem na acessibilidade sua principal ferramenta.

A aplicação do método envolve a edição de arquivos GTFS, o cálculo de matrizes de custo, a tomada de decisão por trás da escolha de qual medida de acessibilidade utilizar, a estimativa dos níveis de acessibilidade, a visualização espacial desses níveis e o cálculo e análise de indicadores de desigualdade. Engloba, portanto, muitos dos pontos abordados neste livro, e serve como uma aplicação prática dos conceitos até então apresentados.

[RAFA: mencionar que uma avaliação de um projeto ou política deve ser a mais abrangente possível, considerando desde aspectos de participação social no desenho do projeto até o seu impacto em termos mais amplos considerados aspectos ambientais, econômicos, sociais, etc. A análise de impacto de acessibilidade, da qual tratamos nesse capítulo, é uma dessas dimensões de impacto.

6 Comparando a acessibilidade entre dois cenários de transporte

Neste capítulo nós avaliaremos o impacto de uma política relativamente simples, mas que pode trazer significativos benefícios em termos de acessibilidade à população: o aumento da frequência das linhas de transporte público.

Para isso, nós precisaremos comparar os níveis de acessibilidade antes e depois da suposta implementação desta política. Precisamos, portanto, modificar o GTFS original da área de estudo para atualizar as frequências das rotas nele descritas, calcular duas matrizes de tempo de viagem distintas (uma antes e outra depois da política), calcular dois conjuntos de níveis de acessibilidade distintos e comparar esse conjunto. Neste capítulo, vamos cobrir este passo-a-passo em detalhes, começando primeiro pela edição do GTFS original.

6.1 Modificação do arquivo GTFS

Nesta análise, nós usaremos novamente a amostra do GTFS da SPTrans, disponível nos pacotes `r5r` e `gtfstools`. Para simularmos um aumento na frequência das viagens de transporte público, nós precisamos diminuir o *headway* de cada viagem. Como este *feed* possui uma tabela `frequencies`, esta edição é particularmente simples: nós precisamos apenas modificar a coluna `headway_secs`. Assumindo uma diminuição de 25% no tempo entre viagens de uma mesma linha, esta manipulação se resume a apenas uma linha de código:

```
library(gtfstools)

path <- system.file("extdata/spo/spo.zip", package = "r5r")

before_gtfs <- read_gtfs(path)
after_gtfs <- read_gtfs(path)

after_gtfs$frequencies[, headway_secs := headway_secs * 0.75]

head(after_gtfs$frequencies)
```


	trip_id	start_time	end_time	headway_secs
1:	CPTM L07-0	04:00:00	04:59:00	540
2:	CPTM L07-0	05:00:00	05:59:00	270
3:	CPTM L07-0	06:00:00	06:59:00	270
4:	CPTM L07-0	07:00:00	07:59:00	270
5:	CPTM L07-0	08:00:00	08:59:00	270
6:	CPTM L07-0	09:00:00	09:59:00	360

```
head(before_gtfs$frequencies)
```

	trip_id	start_time	end_time	headway_secs
1:	CPTM L07-0	04:00:00	04:59:00	720
2:	CPTM L07-0	05:00:00	05:59:00	360
3:	CPTM L07-0	06:00:00	06:59:00	360
4:	CPTM L07-0	07:00:00	07:59:00	360
5:	CPTM L07-0	08:00:00	08:59:00	360
6:	CPTM L07-0	09:00:00	09:59:00	480

Pronto, isso é tudo que precisamos editar no GTFS para simular o impacto de um possível aumento na frequência das viagens de transporte público. É claro que na vida real uma proposta do tipo deve ser acompanhada de estudos de viabilidade financeira, por exemplo, para atestar sua robustez. Para fins puramente demonstrativos, no entanto, esta pequena edição já satisfaz as nossas necessidades.

6.2 Cálculo das matrizes de tempo de viagem

Tendo feito a modificação necessária no nosso GTFS, nós precisamos agora utilizá-lo para calcular a matriz de tempo de viagem, que por sua vez vamos usar para estimar os níveis de acessibilidade. Como apresentado na Section 3.1, para isso nós vamos utilizar a função `travel_time_matrix()`, do pacote `r5r`.

Antes de calcular as matrizes de fato, nós precisamos organizar os nossos arquivos na estrutura que o `r5r` requer. Cada um dos nossos cenários (antes e depois) deve ser representado por uma pasta contendo os arquivos necessários para o roteamento. Vamos criar uma pasta temporária dentro da qual vamos botar as pastas do roteamento:

```
tmpdir <- tempfile("impact_analysis")

dir.create(tmpdir)
```

```
fs::dir_tree(tmpdir)
```

```
/tmp/RtmpeGLQPj/impact_analysis2f2c15ad4f76
```

Dentro desta pasta, vamos criar as nossas pastas de roteamento. Dentro de cada uma elas, por sua vez, vamos salvar os respectivos arquivos GTFS:

```
before_dir <- file.path(tmpdir, "before")
after_dir <- file.path(tmpdir, "after")

dir.create(before_dir)
dir.create(after_dir)

write_gtfs(before_gtfs, path = file.path(before_dir, "before_gtfs.zip"))
write_gtfs(after_gtfs, path = file.path(after_dir, "after_gtfs.zip"))

fs::dir_tree(tmpdir)
```

```
/tmp/RtmpeGLQPj/impact_analysis2f2c15ad4f76
```

```
+-- after
|   \-- after_gtfs.zip
\-- before
     \-- before_gtfs.zip
```

Para estimarmos o tempo de viagem na nossa área de estudo, precisamos ainda de um arquivo representando a rede viária do local, em formato `.pbf`. Este arquivo também é disponibilizado pelo `r5r`. Partindo do pressuposto que uma maior oferta de transporte público não afetaria o traçado das ruas e calçadas na região, e que, portanto a rede viária permaneceria inalterada entre os dois cenários, podemos usar o mesmo arquivo nas duas matrizes de tempo de viagem:

```
pbf_path <- system.file("extdata/spo/spo_osm.pbf", package = "r5r")

file.copy(from = pbf_path, to = file.path(before_dir, "spo_osm.pbf"))
```

```
[1] TRUE
```

```
file.copy(from = pbf_path, to = file.path(after_dir, "spo_osm.pbf"))
```

```
[1] TRUE
```

```
fs::dir_tree(tmpdir)

/tmp/RtmpeGLQPj/impact_analysis2f2c15ad4f76
+-- after
|   +-- after_gtfs.zip
|   \-- spo_osm.pbf
\-- before
    +-- before_gtfs.zip
    \-- spo_osm.pbf
```

Finalmente, podemos começar o cálculo das matrizes. A primeira etapa é construir a rede de transporte multimodal usada pelo `r5r` no roteamento e criar uma conexão com o R5, o que é feito com o comando `setup_r5()`:

```
options(java.parameters = "-Xmx2G")

library(r5r)

my_setup_r5 <- function(path) {
  setup_r5(path, verbose = FALSE)
}

before_r5r_core <- my_setup_r5(before_dir)
after_r5r_core <- my_setup_r5(after_dir)
```

Tendo sido criada a conexão com o R5 para os dois cenários, prosseguimos agora para o cálculo da matriz, de fato. De forma a avaliar a hipotética política adequadamente, nós precisamos usar os mesmos parâmetros em ambos os cenários. Vamos considerar viagens a pé ou por transporte público, permitir até 800 metros nas pernas de acesso e egresso a paradas e limitar o tempo máximo de viagem em até 120 minutos. Vamos considerar o horário de partida de 14:00 em uma típica segunda-feira de operação. Como origens e destinos, vamos usar os centróides da grade disponibilizada no `r5r`:

```
library(data.table)

points <- fread(system.file("extdata/spo/spo_hexgrid.csv", package = "r5r"))

before_ttm <- travel_time_matrix(
  before_r5r_core,
  origins = points,
  destinations = points,
```

```

mode = c("WALK", "TRANSIT"),
departure_datetime = as.POSIXct(
  "13-05-2019 14:00:00",
  format = "%d-%m-%Y %H:%M:%S"
),
max_walk_dist = 800,
max_trip_duration = 120,
verbose = FALSE,
progress = FALSE
)

after_ttm <- travel_time_matrix(
  after_r5r_core,
  origins = points,
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  departure_datetime = as.POSIXct(
    "13-05-2019 14:00:00",
    format = "%d-%m-%Y %H:%M:%S"
  ),
  max_walk_dist = 800,
  max_trip_duration = 120,
  verbose = FALSE,
  progress = FALSE
)

before_ttm

      fromId      toId travel_time
1: 89a8100c603ffff 89a8100c603ffff      0
2: 89a8100c603ffff 89a8100c617ffff     19
3: 89a8100c603ffff 89a8100c60fffff      6
4: 89a8100c603ffff 89a8100c607ffff     11
5: 89a8100c603ffff 89a8100c6abffff     26
---
48166: 89a8100c8dbffff 89a8100c8cbffff      6
48167: 89a8100c8cbffff 89a8100ceaffffff     14
48168: 89a8100c8cbffff 89a8100ce37ffff      9
48169: 89a8100c8cbffff 89a8100c8dbffff      6
48170: 89a8100c8cbffff 89a8100c8cbffff      0

```

after_ttm

	fromId	toId	travel_time
1:	89a8100c603ffff	89a8100c603ffff	0
2:	89a8100c603ffff	89a8100c617ffff	18
3:	89a8100c603ffff	89a8100c60fffff	6
4:	89a8100c603ffff	89a8100c607ffff	11
5:	89a8100c603ffff	89a8100c6abffff	32

48166:	89a8100c8dbffff	89a8100c8cbffff	6
48167:	89a8100c8cbffff	89a8100ceafffff	14
48168:	89a8100c8cbffff	89a8100ce37ffff	9
48169:	89a8100c8cbffff	89a8100c8dbffff	6
48170:	89a8100c8cbffff	89a8100c8cbffff	0

Como podemos ver, alguns tempos de viagem do cenário pós aumento das frequências são menores do que antes da mudança. Maior frequência de transporte público leva a menores tempos de espera nas paradas de embarque e em eventuais transferências, o que leva a menores tempos de viagem. Muitos tempos, no entanto, são iguais nos dois cenários. Isto é uma provável evidência de que as viagens, nesses casos, são feitas a pé, e consequentemente não são afetadas por mudanças no transporte público.

6.3 Cálculo da acessibilidade nos cenários antes e depois

FDSAFSAD

6.4 Visualização dos níveis e da diferença de acessibilidade

FSADFSAD

6.5 Distribuição da acessibilidade entre grupos sociais

FASDFSDA

Part V

SEÇÃO: 5: Dados do Projeto AOP

Objetivo: o objetivo deste capítulo é mostrar como fazer download e analisar os dados do projeto Acesso a Oportunidades (AOP) utilizando o pacote `aopdata` no R.

Nos capítulos anteriores, você aprendeu como calcular indicadores de acessibilidade. No entanto, em muitos casos, você não tem disponibilidade para calcular esses indicadores por conta própria e tudo o que você quer é analisar os resultados que já foram calculados por alguém.

O projeto [Acesso a Oportunidades](#) disponibiliza uma extensa base de dados com informações sobre a distribuição da população, atividades econômicas e serviços públicos, além de várias estimativas de acessibilidade urbana para diversos tipos de atividades. Essas estimativas de acessibilidade estão disponíveis para diferentes modos de transporte (caminhada, bicicleta, transporte público e automóvel), horários do dia (pico e fora-pico), grupos populacionais (segundo níveis de renda, cor, sexo e idade) e para diferentes atividades (empregos, escolas, serviços de saúde e centros de assistência social). Nesta versão, a base de dados traz essas informações para diversos anos (2017, 2018 e 2019), se apoiando em uma única metodologia consistente para as 20 maiores cidades do Brasil. Veja abaixo `?@tbl-tabela_dados_pop` e `?@tbl-tabela_dados_access`.

i As metodologias utilizadas para gerar estes dados são apresentadas em detalhe em publicações separadas, para os dados populacionais e de uso do solo (`pereira2022usosolo?`), e para os dados de acessibilidade (`pereira2022acessibilidade?`).

Part VI

Quais dados estão disponíveis?

Escopo dos dados:

Dados de população, empregos e serviços públicos:

Tabela x. Informações socioeconômicas da população e de distribuição espacial de atividades, segundo ano e fonte de dados.

Dado	Informações
Características sociodemográficas da população	Quantidade de pessoas segundo sexo, faixa de idade e cor/raça
Estabelecimentos de educação	Quantidade de creches e escolas públicas segundo nível infantil
Estabelecimentos de saúde	Quantidade de estabelecimentos de saúde que atendem pelo SUS
Atividade econômica	Quantidade de empregos formais conforme o nível de instrução
Estabelecimentos de assistência social	Quantidade de CRAS

: Informações socioeconômicas da população e de distribuição espacial de atividades, segundo ano e fonte de dados {#tbl-tabela_dados_pop}

Dados de acessibilidade urbana:

Indicador (código)	Descrição
Tempo mínimo de viagem (TMI)	Tempo até a oportunidade mais próxima
Medida cumulativa ativa (CMA)	Quantidade de oportunidades acessíveis em um determinado limite de tempo
Medida cumulativa passiva (CMP)	Quantidade de pessoas que acessam a localidade em um determinado limite de tempo

: Indicadores de acessibilidade calculados no Projeto Acesso a Oportunidades {#tbl-tabela_dados_access}

Todas as bases de dados criadas pelo Projeto Acesso a Oportunidades (AOP) estão disponíveis para download no [site do projeto](#) ou pelo pacote de R `aopdata`. Nas próximas seções são apresentados exemplos de como baixar e visualizar esses dados em R.

7 Dados de população e socioeconômicos

7.1 Download dos dados

Para fazer o download dos dados do projeto AOP usando o pacote `aopdata`, você pode usar a função `read_population()`. Essa função baixa estimativas do Censo Demográfico de 2010 do IBGE sobre a distribuição espacial da população e suas características em termos de renda domiciliar per capita, cor, sexo e idade. Nesta função, o parâmetro `city` permite você indicar os dados de qual cidade serão baixados.

Os dados estão agregados espacialmente em uma grade de [hexágonos H3](#) na resolução 9, na qual cada hexágono tem uma área de 0.11 km², o que equivale a aproximadamente o tamanho de um quarteirão. Para baixar os dados com as informações espaciais de geometria da grade espacial, você deve usar o parâmetro `geometry = TRUE`.

Neste exemplo, abaixo, nós mostramos como baixar os dados de população do Censo de 2010 para Fortaleza.

```
# load libraries
library(aopdata)

# download aop population data
df <- read_population(city='Fortaleza',
                      year=2010,
                      geometry = TRUE,
                      showProgress = FALSE)
```

Os dados da tabela tem essa aparência aqui:

```
head(df)
```

Simple feature collection with 6 features and 22 fields

Geometry type: POLYGON

Dimension: XY

Bounding box: xmin: -38.50828 ymin: -3.889301 xmax: -38.4983 ymax: -3.878958

Geodetic CRS: WGS 84

	year	id_hex	abbrev_muni	name_muni	code_muni	P001	P002	P003	P004	P005
1	2010	89801040323fffff		for Fortaleza	2304400	30	8	21	0	1
2	2010	89801040327fffff		for Fortaleza	2304400	318	77	233	0	8
3	2010	8980104032bfffff		for Fortaleza	2304400	0	0	0	0	0
4	2010	8980104032fffff		for Fortaleza	2304400	103	24	77	0	2
5	2010	89801040333fffff		for Fortaleza	2304400	43	11	31	0	1
6	2010	89801040337fffff		for Fortaleza	2304400	348	86	252	0	10

	P006	P007	P010	P011	P012	P013	P014	P015	P016	R001	R002	R003
1	17	13	3	4	2	3	8	9	1	168.6	1	1
2	168	150	30	50	26	38	80	82	12	202.6	1	1
3	0	0	0	0	0	0	0	0	0	NA	NA	NA
4	53	50	10	16	8	13	25	27	4	245.6	1	1
5	22	21	4	7	3	5	11	12	1	187.3	1	1
6	175	173	34	54	27	41	89	89	14	168.6	1	1


```

geometry
1 POLYGON ((-38.50232 -3.8858...
2 POLYGON ((-38.50527 -3.8840...
3 POLYGON ((-38.49932 -3.8841...
4 POLYGON ((-38.50227 -3.8824...
5 POLYGON ((-38.50237 -3.8893...
6 POLYGON ((-38.50532 -3.8875...

```

De imediato, se nota que os nomes das variáveis (colunas) da base de dados estão organizadas com códigos, como P001, P002...R001, R002 etc. A [descrição completa do dicionário de variáveis está disponível aqui](#). A descrição de algumas dessas colunas é apresentada nas próximas seções, onde mostramos como fazer a visualização de alguns desses dados em mapas e gráficos.

7.2 Mapa de população total

Antes de visualizar os dados do `aopdata`, nós precisamos carregar algumas bibliotecas de visualização e manipulação de dados.

```

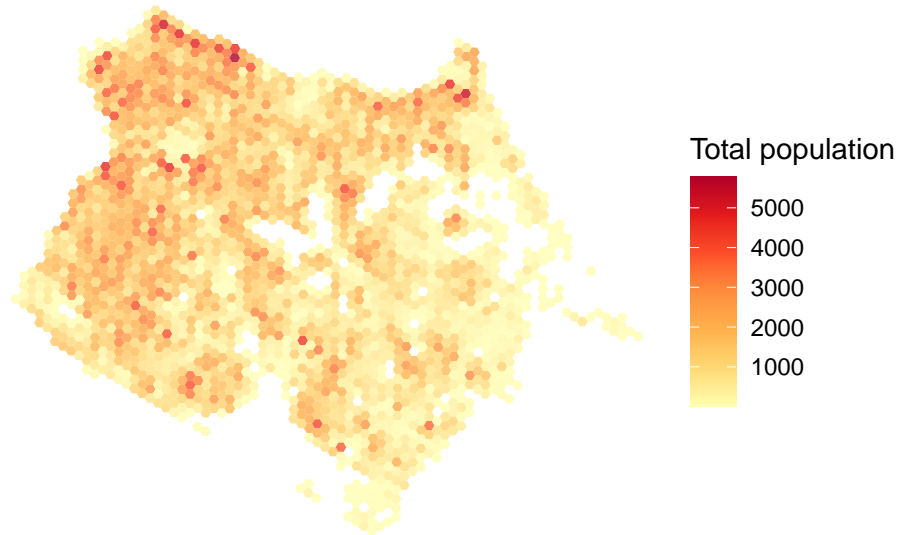
# load libraries
library(patchwork)
library(ggplot2)
library(scales)
library(sf)

```

Com um comando, é possível visualizar a distribuição espacial da população de Fortaleza. A figura mostra um mapa coroplético onde a cor de cada célula da grade espacial é preenchida com base na quantidade total de pessoas residentes (variável P001).

```
ggplot() +  
  geom_sf(data=subset(df, P001>0), aes(fill=P001), color=NA, alpha=.8) +  
  scale_fill_distiller(palette = "YlOrRd", direction = 1)+  
  labs(title='Population distribution',  
        subtitle = 'Fortaleza', fill="Total population") +  
  theme_void()
```

Population distribution
Fortaleza



7.3 Mapa de população por cor

Além da informação sobre a população total em cada célula, o dados do `aopdata` também permitem saber a quantidade de pessoas de diferentes cores (variáveis P002 a P005), sexo (variáveis P006 e P007) e faixas etárias (variáveis P010 à P016). O código abaixo ilustra como é simples calcular a proporção de pessoas negras e brancas em cada hexágono e visualizar esses dados num mapa.

```
pop_b <- ggplot() +
  geom_sf(data=subset(df, P001 >0), aes(fill=P003 / P001), color=NA, alpha=.8) +
  scale_fill_distiller(palette = "RdPu", direction = 1, labels = percent, limits=c(0, 1))+
  labs(title='Proportion of black population', fill="Black pop.") +
  theme_void()

pop_w <- ggplot() +
  geom_sf(data=subset(df, P001 >0), aes(fill=P002 / P001), color=NA, alpha=.8) +
  scale_fill_distiller(palette = "YlGnBu", direction = 1, labels = percent, limits=c(0, 1))+
  labs(title='Proportion of white population', fill="White pop.") +
  theme_void()

# plot figure
pop_b + pop_w
```



7.4 Mapa de população por renda

Os dados trazem também informação sobre a renda domiciliar per capita média de cada hexágono (R001), e sua classificação em termos de quintil (R002) e decil de renda (R003). Com esses dados, é possível visualizar com o comando abaixo a distribuição espacial dos diferentes níveis de renda da cidade.R

```

renda_c <- ggplot() +
  geom_sf(data=subset(df, P001 >0), aes(fill=R001), color=NA, alpha=.8) +
  scale_fill_distiller(palette = "YlOrRd", direction = 1)+
  labs(title='Renda domiciliar per capita média', fill="Renda em R$") +
  theme_void()

renda_d <- ggplot() +
  geom_sf(data=subset(df, !is.na(R002)), aes(fill=factor(R003)), color=NA, alpha=.8) +
  scale_fill_brewer(palette = "RdBu") +
  labs(title='Decils de renda domiciliar per capita', fill="Decil de renda") +
  theme_void() +
  theme(legend.key.size = unit(.3, 'cm'))

# plot figure
renda_c + renda_d

```



8 Dados de distribuição espacial de oportunidades

8.1 Download dos dados

O pacote `aopdata` também permite baixar, para todas cidades incluídas no projeto, dados anuais da distribuição espacial de empregos (de baixa, média e alta escolaridade), estabelecimentos de saúde (de baixa, média e alta complexidade), escolas pública (ensino infantil, fundamental e médio), e de centros de referência de assistência social (Cras).

Todos esses dados podem ser baixados com a função `read_landuse()`, que funciona da mesma maneira que a função `read_population()`. Você só precisa indicar nos parâmetros da função qual cidade (`city`) e ano (`year`) devem ser baixados, além de apontar se deseja que os dados contenham as informações espaciais dos hexágonos (`geometry = TRUE`).

Neste exemplo, abaixo, nós mostramos como baixar os dados de uso do solo no ano de 2019 para Belo Horizonte. Note que essa função automaticamente já baixa também os dados de população, automaticamente.

```
library(aopdata)

# download aop land use data
df <- read_landuse(city='Belo Horizonte',
                  year=2019,
                  geometry = TRUE,
                  showProgress = FALSE)
```

Downloading land use data for the year 2019

Downloading population data for the year 2010

```
head(df)
```

Simple feature collection with 6 features and 39 fields

Geometry type: POLYGON

Dimension: XY

Bounding box: xmin: -43.87914 ymin: -19.86084 xmax: -43.85906 ymax: -19.82421

Geodetic CRS: WGS 84

	id_hex	abbrev_muni	name_muni	code_muni	P001	P002	P003	P004	P005						
1	89a881345a3ffff	bho	Belo Horizonte	3106200	0	0	0	0	0						
2	89a881345a7ffff	bho	Belo Horizonte	3106200	0	0	0	0	0						
3	89a881345b7ffff	bho	Belo Horizonte	3106200	0	0	0	0	0						
4	89a88136103ffff	bho	Belo Horizonte	3106200	508	141	353	4	10						
5	89a88136107ffff	bho	Belo Horizonte	3106200	428	114	304	3	7						
6	89a8813610bffff	bho	Belo Horizonte	3106200	399	111	277	3	8						
	P006	P007	P010	P011	P012	P013	P014	P015	P016	R001	R002	R003	year	T001	T002
1	0	0	0	0	0	0	0	0	0	NA	NA	NA	2019	0	0
2	0	0	0	0	0	0	0	0	0	NA	NA	NA	2019	0	0
3	0	0	0	0	0	0	0	0	0	NA	NA	NA	2019	0	0
4	253	255	53	86	30	43	176	110	10	502.9	2	3	2019	100	17
5	210	218	42	72	27	38	142	98	9	491.8	2	3	2019	44	19
6	203	196	42	68	24	34	138	86	7	502.8	2	3	2019	104	31
	T003	T004	E001	E002	E003	E004	M001	M002	M003	M004	S001	S002	S003	S004	C001
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	72	11	1	1	0	0	321	321	0	0	0	0	0	0	0
5	23	2	0	0	0	0	0	0	0	0	0	0	0	0	0
6	58	15	1	0	1	0	817	0	817	0	0	0	0	0	0

geometry

```

1 POLYGON ((-43.86011 -19.829...
2 POLYGON ((-43.86313 -19.827...
3 POLYGON ((-43.86321 -19.830...
4 POLYGON ((-43.8731 -19.8608...
5 POLYGON ((-43.87612 -19.859...
6 POLYGON ((-43.87001 -19.859...

```

Antes de visualizar os dados de uso do solo nas próximas seções, vamos carregar algumas bibliotecas de visualização e manipulação de dados.

```

# load libraries
library(patchwork)
library(ggplot2)
library(scales)
library(sf)

```


8.2 Mapa de empregos

Como nos exemplos anteriores, é possível visualizar o mapa de distribuição espacial de empregos usando o pacote `ggplot2` com o código abaixo:

```
ggplot() +  
  geom_sf(data=df, aes(fill=T001), color=NA, alpha=.9) +  
  scale_fill_distiller(palette = "YlGnBu", direction = 1) +  
  labs(title='Distribuição espacial de empregos',  
        subtitle = 'Belo Horizonte', fill="N. de empregos") +  
  theme_void()
```

Distribuição espacial de empregos
Belo Horizonte



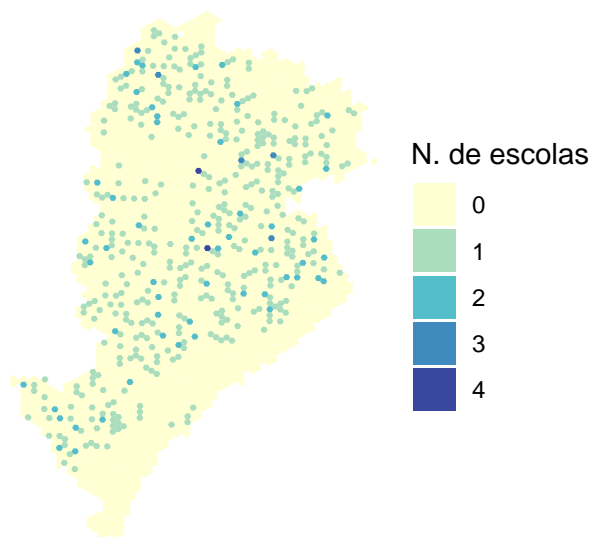
8.3 Mapa de escolas

As variáveis que indicam o número de escolas em cada célula são aquelas que começam com a letra `E__`. Neste exemplo abaixo, nós mapeamos a distribuição espacial de todas as escolas de Belo Horizonte a partir da variável `E001`.

```
ggplot() +  
  geom_sf(data=df, aes(fill=as.factor(E001)), color=NA, alpha=.9) +
```

```
scale_fill_brewer(palette = "YlGnBu", direction = 1) +
labs(title='Distribuição espacial de escolas',
      subtitle = 'Belo Horizonte', fill="N. de escolas") +
theme_void()
```

Distribuição espacial de escolas
Belo Horizonte



8.4 Mapa de estabelecimentos de saúde

Para analisar a distribuição espacial de estabelecimentos de saúde, você deve analisar as variáveis que começam com a letra S___. O código abaixo nos permite comparar a distribuição espacial de estabelecimentos de saúde de baixa complexidade (S002) e alta complexidade (S004).

```
sau_b <- ggplot() +
  geom_sf(data=df, aes(fill=as.factor(S002)), color=NA, alpha=.9) +
  scale_fill_brewer(palette = "YlGnBu", direction = 1, limits=factor(0:4)) +
  labs(title='Estabelecimentos de saúde', subtitle = 'Baixa complexidade',
        fill="N. de estabelecimentos") +
  theme_void()

sau_a <- ggplot() +
```

```

geom_sf(data=df, aes(fill=as.factor(S004)), color=NA, alpha=.9) +
scale_fill_brewer(palette = "YlGnBu", direction = 1, limits=factor(0:4)) +
labs(title='Estabelecimentos de saúde', subtitle = 'Alta complexidade',
      fill="N. de estabelecimentos") +
theme_void()

# plot maps
sau_b + sau_a + plot_layout(guides = 'collect')

```



8.5 Mapa de CRAS

Por fim, a distribuição espacial dos Centros de Referência de Assistência Social (Cras) pode ser analisada com a variável C001.

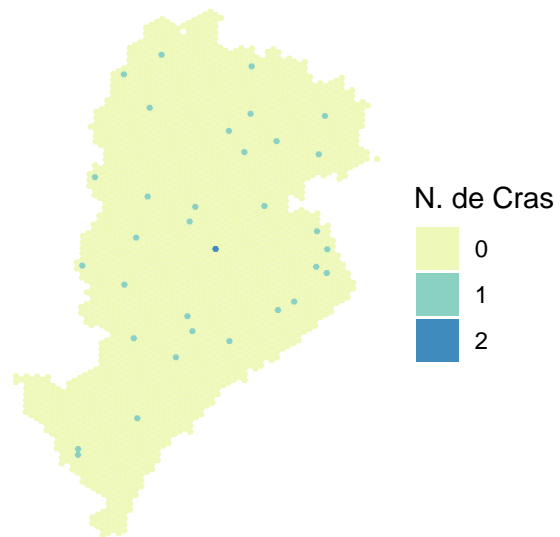
```

ggplot() +
geom_sf(data=df, aes(fill=as.factor(C001)), color=NA, alpha=.9) +
scale_fill_brewer(palette = "YlGnBu", direction = 1) +
labs(title='Centros de Referência de Assistência Social (Cras)',
      subtitle = 'Belo Horizonte', fill="N. de Cras") +
theme_void()

```

Centros de Referência de Assistência Social (Cras)

Belo Horizonte



9 Estimativas e mapas de acessibilidade

9.1 Download dos dados

Finalmente, o pacote `aopdata` também permite baixar, para todas cidades incluídas no projeto, estimativas anuais de acesso a empregos, serviços saúde, educação e assistência social por modo de transporte

Todos esses dados podem ser baixados com a função `read_access()`, que funciona da mesma maneira que as funções `read_population()` e `read_landuse()` apresentadas nos capítulos anteriores. Aqui, no entanto, além de indicar a cidade (`city`) e o ano (`year`) de referência para baixar os dados, você também precisa informar qual o modo de transporte (`mode`) será baixado e se você quer estimativas de acessibilidade no horário de pico (`peak = TRUE`) ou fora-pico (`peak = FALSE`). Esses dados representam a acessibilidade mediana do período de pico (entre 6h e 8h) e fora-pico (entre 14h e 16h).

Neste exemplo, abaixo, nós mostramos como baixar os dados de uso de acessibilidade urbana no ano de 2019 para São Paulo no período de pico. Nesse exemplo, nós baixamos tanto as estimativas de acessibilidade por automóvel quanto por transporte público, e empilhamos os dados num único `data.frame`. Note que essa função automaticamente já baixa também os dados de população e de uso do solo, automaticamente.

```
library(aopdata)

# download aop accessibility data
df_pt <- read_access(
  city='São Paulo',
  mode='public_transport',
  year=2019,
  peak = TRUE,
  geometry = TRUE,
  showProgress = FALSE
)

df_car <- read_access(
  city='São Paulo',
```

```

mode='car',
year=2019,
peak = TRUE,
geometry = TRUE,
showProgress = FALSE
)

# row bind into a single data.frame
df <- rbind(df_pt, df_car)
head(df)

```

Simple feature collection with 6 features and 205 fields

Geometry type: POLYGON

Dimension: XY

Bounding box: xmin: -46.63863 ymin: -23.71413 xmax: -46.62834 ymax: -23.70485

Geodetic CRS: WGS 84

	id_hex	abbrev_muni	name_muni	code_muni	year	P001	P002	P003	P004	P005
1	89a81000003ffff	spo	Sao Paulo	3550308	2019	322	127	190	0	5
2	89a81000007ffff	spo	Sao Paulo	3550308	2019	16	3	13	0	0
3	89a8100000bffff	spo	Sao Paulo	3550308	2019	2386	1142	1232	2	10
4	89a8100000fffff	spo	Sao Paulo	3550308	2019	885	260	622	0	3
5	89a81000013ffff	spo	Sao Paulo	3550308	2019	725	340	380	0	5
6	89a81000017ffff	spo	Sao Paulo	3550308	2019	211	110	98	0	3

	P006	P007	P010	P011	P012	P013	P014	P015	P016	R001	R002	R003	T001	T002	T003
1	158	164	33	63	28	35	85	75	3	477.6	2	3	72	19	50
2	9	7	2	2	2	3	2	5	0	65.7	1	1	0	0	0
3	1216	1170	247	410	174	269	682	572	32	377.2	1	1	52	6	24
4	460	425	96	124	88	127	175	266	9	363.9	1	1	0	0	0
5	371	354	85	129	51	78	196	173	13	503.5	2	3	0	0	0
6	104	107	20	33	16	25	52	59	6	687.9	3	6	113	18	65

	T004	E001	E002	E003	E004	M001	M002	M003	M004	S001	S002	S003	S004	C001
1	3	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	22	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	30	2	0	2	1	1477	0	1168	309	0	0	0	0	0

	mode	peak	CMATT15	CMATB15	CMATM15	CMATA15	CMAST15	CMASB15	CMASM15
1	public_transport	1	315	71	175	69	0	0	0
2	public_transport	1	313	70	174	69	0	0	0
3	public_transport	1	229	43	161	25	2	2	2
4	public_transport	1	551	123	343	85	3	3	3

5	public_transport	1	421	90	262	69	2	2	2
6	public_transport	1	312	76	187	49	0	0	0
	CMASA15	CMAET15	CMAEI15	CMAEF15	CMAEM15	CMAMT15	CMAMI15	CMAMF15	CMAMM15
1	0	2	0	2	1	1477	0	1168	309
2	0	2	0	2	1	1477	0	1168	309
3	0	0	0	0	0	0	0	0	0
4	0	4	1	3	2	3132	460	1965	707
5	0	2	0	2	1	1477	0	1168	309
6	0	2	0	2	1	1477	0	1168	309
	CMACT15	CMPPPT15	CMPPPH15	CMPPPM15	CMPPPB15	CMPPA15	CMPPPI15	CMPPN15	CMPP0005I15
1	0	4895	2428	2467	2238	27	2	2628	505
2	0	4895	2428	2467	2238	27	2	2628	505
3	0	9537	4645	4892	4685	64	3	4785	872
4	0	13109	6331	6778	6453	61	8	6587	1088
5	0	4934	2452	2482	2368	48	0	2518	461
6	0	3199	1608	1591	1567	20	0	1612	317
	CMPP0614I15	CMPP1518I15	CMPP1924I15	CMPP2539I15	CMPP4069I15	CMPP70I15	CMATT30		
1	809		371	566	1313	1247	84	4797	
2	809		371	566	1313	1247	84	4228	
3	1538		726	1138	2550	2539	174	2648	
4	2012		967	1575	3383	3783	301	6200	
5	749		373	587	1271	1372	121	5773	
6	487		239	374	807	891	84	6608	
	CMATB30	CMATM30	CMATA30	CMAST30	CMASB30	CMASM30	CMASA30	CMAET30	CMAEI30
1	1293	2789	715	5	5	5	0	18	5
2	1170	2452	606	4	4	4	0	13	4
3	714	1526	408	4	4	4	0	9	2
4	1598	3641	961	5	5	5	0	19	6
5	1589	3271	913	5	5	5	0	18	5
6	1821	3659	1128	6	6	6	0	22	5
	CMAEF30	CMAEM30	CMAMT30	CMAMI30	CMAMF30	CMAMM30	CMACT30	CMPPPT30	CMPPPH30
1	13	4	15148	2409	10765	1974	1	59041	28457
2	9	3	10497	1949	6931	1617	1	40049	19392
3	7	3	7222	1069	5137	1016	1	59792	28974
4	13	5	15710	2527	10551	2632	2	101380	48709
5	13	4	15148	2409	10765	1974	1	81985	39473
6	17	5	19547	2409	14343	2795	1	83186	40111
	CMPPPM30	CMPPPB30	CMPPA30	CMPPPI30	CMPPN30	CMPP0005I30	CMPP0614I30	CMPP1518I30	
1	30584	28535	446	29	30031	5263	8809	4111	
2	20657	19159	273	14	20603	3610	6061	2866	
3	30818	28424	365	25	30978	5376	9204	4241	
4	52671	47601	955	66	52758	9076	15239	6971	
5	42512	39784	695	49	41457	7189	12278	5675	

6	43075	40359	728	51	42048	7302	12371	5749	
	CMPP1924I30	CMPP2539I30	CMPP4069I30	CMPP70I30	CMATT60	CMATB60	CMATM60	CMATA60	
1	6817	15664	16733	1644	96195	18711	54773	22711	
2	4673	10486	11274	1079	72810	14161	40949	17700	
3	6927	15859	16646	1539	147237	26399	82208	38630	
4	11658	27259	28303	2874	107938	21292	61768	24878	
5	9311	21712	23351	2469	161353	28971	91687	40695	
6	9459	22070	23710	2525	158016	28997	91280	37739	
	CMAST60	CMASB60	CMASM60	CMASA60	CMAET60	CMAEI60	CMAEF60	CMAEM60	CMAMT60
1	50	38	48	6	120	40	76	29	87761
2	40	34	39	2	108	36	68	28	79902
3	57	45	56	5	135	45	88	30	93958
4	56	43	54	6	128	42	82	31	92799
5	59	44	57	7	141	47	90	32	97925
6	56	43	54	6	139	47	87	32	96281
	CMAMI60	CMAMF60	CMAMM60	CMACT60	CMPPT60	CMPPH60	CMPPM60	CMPPB60	CMPPA60
1	15059	56558	16144	5	547471	259971	287500	277734	11134
2	14014	50455	15433	5	501558	238550	263008	247962	8781
3	16318	61918	15722	6	633994	299454	334540	339064	19838
4	15615	60049	17135	5	689942	325560	364382	378571	21810
5	16744	63838	17343	6	666492	314759	351733	360532	20661
6	16812	62265	17204	6	632529	299225	333304	334655	17957
	CMPP160	CMPPN60	CMPP0005I60	CMPP0614I60	CMPP1518I60	CMPP1924I60	CMPP2539I60		
1	407	258196	44866	76195	34899	59393	146870		
2	383	244432	41903	71199	32501	54873	134759		
3	473	274619	49586	83666	38585	67298	169669		
4	496	289065	53223	89680	41446	72354	184045		
5	483	284816	51720	87430	40333	70418	178015		
6	451	279466	49804	84545	38888	67458	168928		
	CMPP4069I60	CMPP70I60	CMATT90	CMATB90	CMATM90	CMATA90	CMAST90	CMASB90	CMASM90
1	162048	23200	1168463	166686	577124	424653	169	111	160
2	146361	19962	1088120	153982	540566	393572	160	107	152
3	194234	30956	1428750	211239	719220	498291	190	119	179
4	214049	35145	1362473	191461	669950	501062	182	117	172
5	205504	33072	1443193	211449	723001	508743	203	131	191
6	192822	30084	1406215	203383	701014	501818	204	132	193
	CMASA90	CMAET90	CMAEI90	CMAEF90	CMAEM90	CMAMT90	CMAMI90	CMAMF90	CMAMM90
1	42	337	107	207	92	218542	30864	131526	56152
2	39	328	105	201	90	213963	30365	128917	54681
3	50	370	123	222	101	232126	34693	138061	59372
4	46	355	117	214	92	223492	32779	134438	56275
5	51	415	136	248	114	264729	38719	158252	67758
6	51	415	137	247	114	266946	39578	159137	68231

	CMACT90	CMPTT90	CMPPH90	CMPPM90	CMPPB90	CMPPA90	CMPP190	CMPPN90	CMPP0005I90
1	10	1697502	790958	906544	1113577	80011	1336	502578	111963
2	9	1617428	753988	863440	1054271	74800	1240	487117	107419
3	10	1918445	894649	1023796	1274988	90290	1602	551565	125137
4	10	2096063	978494	1117569	1384576	94542	2065	614880	138110
5	12	2027043	946208	1080835	1333715	91633	1714	599981	133501
6	12	1950319	910888	1039431	1272271	88334	1558	588156	129637
	CMPP0614I90	CMPP1518I90	CMPP1924I90	CMPP2539I90	CMPP4069I90	CMPP70I90			
1	183412	87698	165449	454122	573567	121291			
2	176240	84154	157827	432825	545306	113657			
3	204140	97829	187004	513061	650643	140631			
4	225800	107835	205069	561408	707435	150406			
5	218843	104726	198411	541774	684520	145268			
6	212862	101535	191408	521731	655761	137385			
	CMATT120	CMATB120	CMATM120	CMATA120	CMAST120	CMASB120	CMASM120	CMASA120	
1	2136716	328724	1094450	713542	357	235	338	82	
2	2095040	321900	1068319	704821	355	235	337	80	
3	2203072	344047	1136019	723006	374	243	356	86	
4	2494268	397258	1308021	788989	418	266	395	97	
5	2275396	357833	1177979	739584	401	265	381	90	
6	2195795	342423	1130056	723316	387	257	367	88	
	CMAET120	CMAEI120	CMAEF120	CMAEM120	CMAMT120	CMAMI120	CMAMF120	CMAMM120	
1	821	288	481	221	518042	83000	313683	121359	
2	815	288	479	216	516846	83610	313657	119579	
3	855	293	508	230	532772	82410	323147	127215	
4	976	340	576	260	610890	97742	369803	143345	
5	948	333	559	251	600516	97066	367244	136206	
6	942	334	554	254	604830	98765	368433	137632	
	CMACT120	CMPTT120	CMPPH120	CMPPM120	CMPPB120	CMPPA120	CMPP1120	CMPPN120	
1	20	4018897	1890513	2128384	2540395	134426	4039	1340037	
2	20	3860887	1814742	2046145	2445536	130803	3868	1280680	
3	21	4320562	2032079	2288483	2745882	142238	4320	1428122	
4	24	4752649	2234731	2517918	3037441	153875	5098	1556235	
5	22	4351102	2049261	2301841	2735091	140022	4377	1471612	
6	21	4191716	1974700	2217016	2627575	136196	4172	1423773	
	CMPP0005I120	CMPP0614I120	CMPP1518I120	CMPP1924I120	CMPP2539I120	CMPP4069I120			
1	281477	467671	217602	408477	1084792	1300130			
2	269676	448086	208699	391944	1041317	1251187			
3	301337	501274	233530	439651	1163931	1400822			
4	330744	549438	256378	484187	1277780	1541189			
5	307070	511736	237911	444278	1173140	1401287			
6	297029	494835	229726	427661	1129908	1348228			
	CMPP70I120	TMIST	TMISB	TMISM	TMISA	TMJET	TMIEI	TMIEF	TMIEI

1	258748	17	17	17	53	8.0	26	8.0	8.0	21
2	249978	21	21	21	55	9.0	28	9.0	9.0	23
3	280017	13	13	13	50	17.0	26	17.0	17.0	30
4	312933	13	13	13	53	7.0	13	7.0	7.0	28
5	275680	12	12	12	51	8.0	24	8.0	8.0	19
6	264329	16	16	16	49	5.8	22	5.8	5.8	16

geometry

```
1 POLYGON ((-46.63251 -23.711...
2 POLYGON ((-46.63552 -23.709...
3 POLYGON ((-46.62941 -23.709...
4 POLYGON ((-46.63242 -23.708...
5 POLYGON ((-46.6326 -23.7141...
6 POLYGON ((-46.63561 -23.712...
```

Os nomes das variáveis (colunas) com estimativas de acessibilidade também estão organizadas com códigos, como CMAEF30, TMISB ou CMPPM60. O nome das colunas com estimativas de acessibilidade são a junção de três componentes: 1) Tipo de indicador de acessibilidade 2) Tipo de oportunidade / pessoas 3) Tempo limite

1) O **tipo de indicador** de acessibilidade é indicado pelas primeiras 3 letras. O projeto AOP atualmente inclui três tipos de indicadores:

- CMA Indicador de acessibilidade cumulativo ativo
- CMP Indicador de acessibilidade cumulativo passivo
- TMI Indicador de tempo mínimo até oportunidade mais próxima

2) O **tipo de atividade** é indicado pelas letras seguintes, no meio do nome da variável. O projeto AOP atualmente inclui diversos tipos de atividades:

- TT Todos empregos
- TB Empregos de baixa escolaridade
- TM Empregos de média escolaridade
- TA Empregos de alta escolaridade
- ST Todos estabelecimentos de saúde
- SB Estabelecimentos de saúde de baixa complexidade
- SM Estabelecimentos de saúde de média complexidade
- SA Estabelecimentos de saúde de alta complexidade
- ... e assim por diante.

No caso do indicador de acessibilidade passiva (CMP), as letras do meio do nome da variável indicam qual o grupo populacional de referência.

- PT População total

- PH População de homens
- PM População de mulheres
- PB População branca
- PN População negra
- P1924I População de 19 a 24 anos de idade
- P2539I População de 25 a 39 anos de idade

3) O **tempo limite de viagem** é indicado pelos números no final do nome da variável. Esses números somente se aplicam para os indicadores de acessibilidade cumulativa ativa (CMA) e passiva (CMP).

Exemplos:

CMAEF30: Número de escolas de ensino fundamental acessíveis em até 30 minutos

TMISB: Tempo de viagem até o estabelecimento de saúde mais próximo com serviços de baixa complexidade

CMPPM60: Quantidade de mulheres que conseguem acessar determinado hexágono em até 60 minutos

Lembre-se, a [descrição completa do dicionário de variáveis está disponível no site do pacote aopdata](#).

A seguir, nós mostramos alguns exemplos de como visualizar essas estimativas de acessibilidade.

9.2 Mapa do tempo para acessar o hospital mais próximo

Neste exemplo, nós vamos comparar o nível de acessibilidade até hospitais entre modos automóvel *vs* transporte público. Para analisar qual o tempo mínimo de viagem (TMI) até um hospital de alta complexidade (SA), nós precisamos analisar a variável TMISA. Com o código abaixo, nós carregamos as bibliotecas para manipulação e visualização de dados, e visualizamos a distribuição espacial dos valores de TMISA para ambos modos de transporte.

Note, no entanto, que os tempos de viagem por transporte público costumam ser muito mais longos do que por automóvel. Então para facilitar a visualização dos resultados, nós truncamos a distribuição dos valores de TMISA em 60 minutos ou mais.

```
# load libraries
library(ggplot2)
library(data.table)
library(patchwork)
library(scales)
```

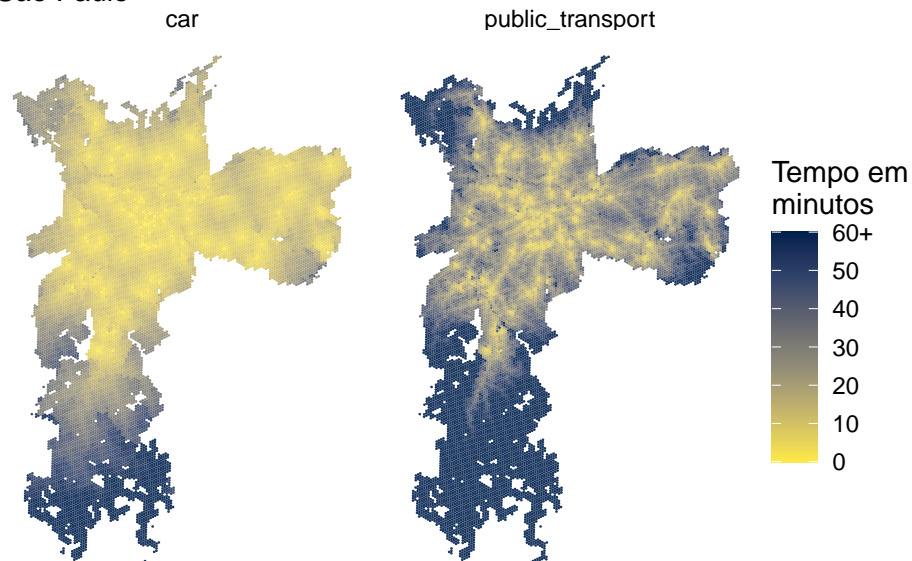
```
library(sf)

# truncate max values to 60 min.
df$TMISA <- ifelse(df$TMISA > 60, 60, df$TMISA)

# plot
ggplot() +
  geom_sf(data=subset(df, !is.na(mode)), aes(fill=TMISA), color=NA, alpha=.9) +
  scale_fill_viridis_c(option = 'cividis', direction = -1,
    breaks = seq(0,60,10),
    labels = c(seq(0,50,10), "60+")) +
  labs(title='Tempo de viagem até hospital de alta complex. mais próximo',
    subtitle = 'São Paulo', fill="Tempo em\nminutos") +
  facet_grid(.~mode) +
  theme_void()
```

Tempo de viagem até hospital de alta complex. mais próximo

São Paulo



9.3 Mapa da quantidade de oportunidades acessíveis

Uma vez que os dados já foram baixados do `aopdata`, é muito simples comparar por exemplo a quantidade de oportunidades acessíveis em diferentes tempos de viagem. O código abaixo

ilustra como visualizar num mapa o número de empregos acessíveis em até 60 e 90 minutos de viagem por transporte público, e como colocar esses mapas lado a lado para comparação.

```
# limits for legend scale
value_limits <- c(0, max(df_pt$CMATT90, na.rm=T)/1000)

# create maps
fig60 <- ggplot() +
  geom_sf(data=subset(df_pt, !is.na(mode)), aes(fill=CMATT60/1000), color=NA, alpha=0.5) +
  scale_fill_viridis_c(option = 'inferno', limits = value_limits) +
  labs(subtitle = 'em até 60 min.', fill="Empregos") +
  theme_void()

fig90 <- ggplot() +
  geom_sf(data=subset(df_pt, !is.na(mode)), aes(fill=CMATT90/1000), color=NA, alpha=0.5) +
  scale_fill_viridis_c(option = 'inferno', limits = value_limits) +
  labs(subtitle = 'em até 90 min.', fill="Empregos") +
  theme_void()

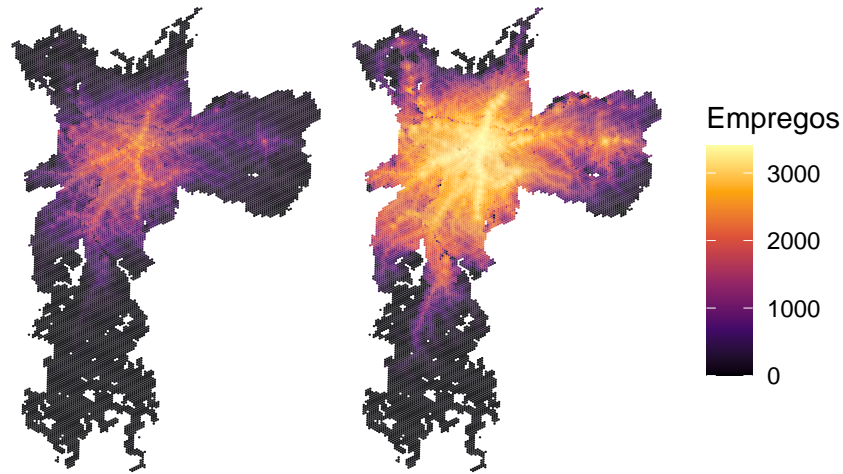
# plot figure
fig60 + fig90 +
  plot_layout(guides = 'collect') +
  plot_annotation(title = 'Quantidade de empregos acessíveis por transporte público',
    subtitle = "São Paulo")
```

Quantidade de empregos acessíveis por transporte público

São Paulo

em até 60 min.

em até 90 min.



9.4 Desigualdades de acesso a oportunidades

Existem diversas maneiras de se analisar quão desiguais são as condições de acesso a oportunidades a partir dos dados do `aopdata`. Nós apresentamos nesta subseção três exemplos de como esse tipo de análise pode ser realizada.

Desigualdade no tempo de acesso TMI

Neste primeiro exemplo, nós vamos comparar qual o tempo médio de viagem até o hospital mais próximo para pessoas de diferentes níveis de renda. Para isso, o código abaixo calcula o valor médio de `TMISA` ponderada pela população em cada hexágono. Essa ponderação é necessária porque o número de hexágonos pode variar muito entre hexágonos.

Antes disso, cabe observar que alguns hexágonos da cidade não conseguem acessar nenhum hospital em até 2h de viagem. Em casos como esse, o valor das variáveis `TMI__` é infinito (`Inf`). Para lidar com esses casos, nós substituímos abaixo todos valores `Inf` por 120 minutos.

```
# copy data to new data.table
dt <- copy(df_pt)
setDT(dt)

# replace Inf travel time with 120
```

```
dt[, TMISA := fifelse(TMISA==Inf, 120, TMISA)]

# calculate average travel time by income
temp <- dt[, .(average = weighted.mean(x=TMISA, w=P001, na.rm=T)), by=R003]
temp <- na.omit(temp)

ggplot() +
  geom_col(data=temp, aes(y=average, x=factor(R003)), fill='#2c9e9e', color=NA) +
  scale_x_discrete(labels=c("D1\n+Pobres", paste0('D', 2:9), "D10\n+Ricos")) +
  labs(title = 'Média de tempo de viagem até o hospital mais proximo',
       subtitle = 'por transporte público em São Paulo',
       x='Decil de renda', y='Tempo de viagem\nem min.') +
  theme_minimal()
```

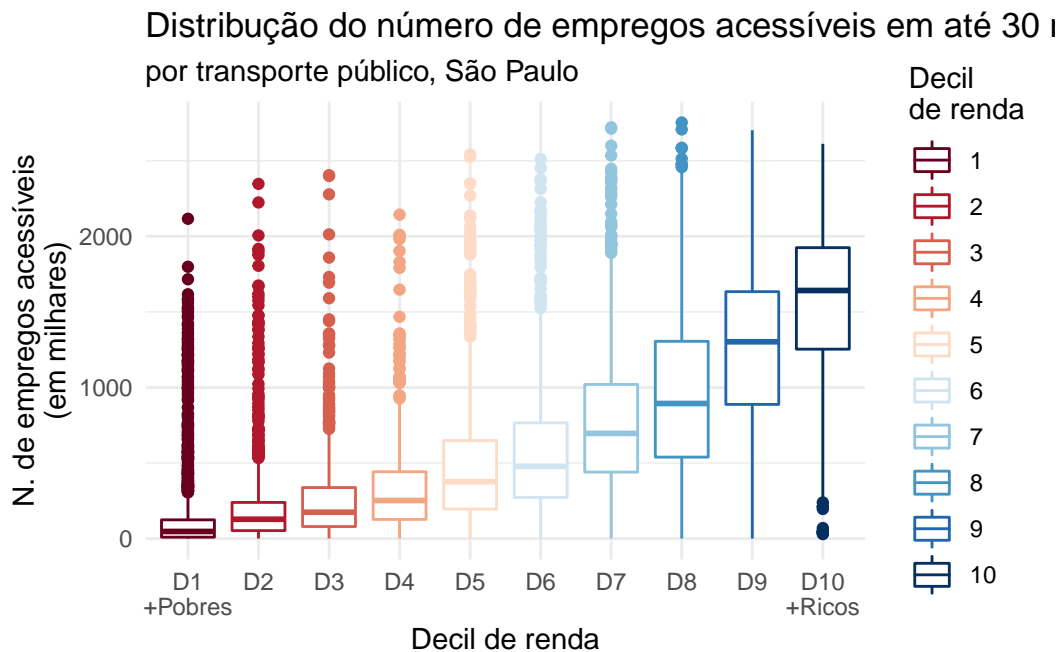


Desigualdade do número de oportunidades acessíveis CMA

Outra maneira de examinar a desigualdade de acesso a oportunidades é comparar a quantidade de oportunidades acessíveis por diferentes grupos populacionais considerando-se um mesmo modo de transporte e tempo de viagem. Nesse caso, nós analisamos o Indicador de acessibilidade cumulativo ativo (CMA).

Neste exemplo abaixo, nós utilizamos *box plots* para comparar a quantidade de empregos acessíveis por transporte público em até 30 minutos de viagem.

```
ggplot() +
  geom_boxplot(data=subset(dt, !is.na(R003)),
               aes(x = factor(R003), y=CMATT60/1000, color=factor(R003))) +
  scale_color_brewer(palette = 'RdBu') +
  labs(title='Distribuição do número de empregos acessíveis em até 30 min.', color="Decil\n",
       subtitle='por transporte público, São Paulo',
       x='Decil de renda', y="N. de empregos acessíveis\n(em milhares)") +
  scale_x_discrete(labels=c("D1\n+Pobres", paste0('D', 2:9), "D10\n+Ricos")) +
  theme_minimal()
```



Também é possível comparar como diferentes modos de transporte possibilitam diferentes níveis de acessibilidade, e como essa diferença pode variar muito entre cidades. Nesse exemplo, abaixo, nós vamos comparar a quantidade de empregos acessíveis em até 30 minutos de viagem a pé e de carro. O primeiro passo é baixar os dados de acessibilidade a pé e por automóvel para todas as cidades.

```
# download data
df_car <- read_access(city = 'all', mode='car', year = 2019, showProgress = FALSE)
df_walk <- read_access(city = 'all', mode='walk', year = 2019, showProgress = FALSE)
```

O próximo passo é calcular para cada cidade a média ponderada do número de empregos acessíveis em até 30 minutos (CMATT30) para cada um dos modos de transporte. Com esses

resultados prontos para cada modo, nós juntamos essas estimativas num único `data.frame` e calculamos a razão da acessibilidade por carro dividida pela acessibilidade à pé.

```
# calculate the average number of jobs accessible in 30min
df_car2 <- df_car[, .(access_car = weighted.mean(CMATT30, w = P001, na.rm=T)), by=name_muni]
df_walk2 <- df_walk[, .(access_walk = weighted.mean(CMATT30, w = P001, na.rm=T)), by=name_muni]

# merge and get the ratio between walking and driving access
df <- merge(df_car2, df_walk2)
df[, ratio := access_car/access_walk]

head(df)
```

	name_muni	access_car	access_walk	ratio
1:	Belem	155270.4	9392.235	16.53179
2:	Belo Horizonte	529890.0	12464.233	42.51284
3:	Brasilia	220575.9	4110.703	53.65892
4:	Campinas	256333.1	6748.923	37.98133
5:	Campo Grande	172680.5	4181.209	41.29919
6:	Curitiba	494376.9	10471.135	47.21331

Pronto, agora é só visualizar os resultados. Como esperado, a figura abaixo mostra que é possível acessar muito mais empregos em 30 min. por automóvel do que em caminhada. No entanto, essa diferença varia muito entre cidades. Em São Paulo e em Brasília, 30 minutos de viagem por automóvel permite acessar, em média, um número de empregos 54 vezes maior do que a pé. Em Belém, onde observamos a menor diferença, o automóvel ainda permite acessar 17 vezes mais empregos do que a caminhada.

```
# plot
ggplot(data=df, aes(x=ratio, y=reorder(name_muni, ratio))) +
  geom_bar(stat='identity') +
  geom_text(aes(x=ratio+6, label=paste0(round(ratio), 'x')) +
  labs(title = 'Diferença da quantidade de empregos acessíveis por\ntautomóvel vs a pé',
        subtitle= 'em até 30 min. de viagem',
        y='', x='Número de vezes') +
  theme_classic()
```

Diferença da quantidade de empregos acessíveis por
automóvel vs a pé
em até 30 min. de viagem



Referências bibliográficas

- Arbex, Renato, and Claudio B. Cunha. 2020. "Estimating the Influence of Crowding and Travel Time Variability on Accessibility to Jobs in a Large Public Transport Network Using Smart Card Big Data." *Journal of Transport Geography* 85 (May): 102671. <https://doi.org/10.1016/j.jtrangeo.2020.102671>.
- Banister, David. 2011. "The Trilogy of Distance, Speed and Time." *Journal of Transport Geography* 19 (4): 950–59. <https://doi.org/10.1016/j.jtrangeo.2010.12.004>.
- Bertolini, L., F. le Clercq, and L. Kapoen. 2005. "Sustainable Accessibility: A Conceptual Framework to Integrate Transport and Land Use Plan-Making. Two Test-Applications in the Netherlands and a Reflection on the Way Forward." *Transport Policy* 12 (3): 207–20. <https://doi.org/10.1016/j.tranpol.2005.01.006>.
- Brodsky, Isaac. 2018. "H3: Uber's Hexagonal Hierarchical Spatial Index." *Uber Engineering Blog*. <https://eng.uber.com/h3/>.
- Church, A, M Frost, and K Sullivan. 2000. "Transport and Social Exclusion in London." *Transport Policy* 7 (3): 195–205. [https://doi.org/10.1016/S0967-070X\(00\)00024-X](https://doi.org/10.1016/S0967-070X(00)00024-X).
- Conway, Matthew Wigginton, Andrew Byrd, and Marco van der Linden. 2017. "Evidence-Based Transit and Land Use Sketch Planning Using Interactive Accessibility Methods on Combined Schedule and Headway-Based Networks." *Transportation Research Record: Journal of the Transportation Research Board* 2653 (1): 45–53. <https://doi.org/10.3141/2653-06>.
- Dijst, Martin, Tom de Jong, and Jan Ritsema van Eck. 2002. "Opportunities for Transport Mode Change: An Exploration of a Disaggregated Approach." *Environment and Planning B: Planning and Design* 29 (3): 413–30. <https://doi.org/10.1068/b12811>.
- Dong, Xiaojing, Moshe E. Ben-Akiva, John L. Bowman, and Joan L. Walker. 2006. "Moving from Trip-Based to Activity-Based Measures of Accessibility." *Transportation Research Part A: Policy and Practice* 40 (2): 163–80. <https://doi.org/10.1016/j.tra.2005.05.002>.
- El-Geneidy, Ahmed, David Levinson, Ehab Diab, Genevieve Boisjoly, David Verbich, and Charis Loong. 2016. "The Cost of Equity: Assessing Transit Accessibility and Social Disparity Using Total Travel Cost." *Transportation Research Part A: Policy and Practice* 91 (September): 302–16. <https://doi.org/10.1016/j.tra.2016.07.003>.
- Farrington, John, and Conor Farrington. 2005. "Rural Accessibility, Social Inclusion and Social Justice: Towards Conceptualisation." *Journal of Transport Geography* 13 (1): 1–12. <https://doi.org/10.1016/j.jtrangeo.2004.10.002>.
- Geurs, Karst, and Bert van Wee. 2004. "Accessibility Evaluation of Land-Use and Transport Strategies: Review and Research Directions." *Journal of Transport Geography* 12 (2): 127–40. <https://doi.org/10.1016/j.jtrangeo.2003.10.005>.

- Herszenhut, Daniel, Rafael H. M. Pereira, Licinio da Silva Portugal, and Matheus Henrique de Sousa Oliveira. 2022. "The Impact of Transit Monetary Costs on Transport Inequality." *Journal of Transport Geography* 99 (February): 103309. <https://doi.org/10.1016/j.jtrangeo.2022.103309>.
- Higgins, Christopher, Matthew Palm, Amber DeJohn, Luna Xi, James Vaughan, Steven Farber, Michael Widener, and Eric Miller. 2022. "Calculating Place-Based Transit Accessibility: Methods, Tools and Algorithmic Dependence." *Journal of Transport and Land Use* 15 (1). <https://doi.org/10.5198/jtlu.2022.2012>.
- Levinson, David, and David King. 2020. *Transport Access Manual: A Guide for Measuring Connection Between People and Places*. Committee of the Transport Access Manual, University of Sydney.
- Lucas, Karen, Giulio Mattioli, Ersilia Verlinghieri, and Alvaro Guzman. 2016. "Transport Poverty and Its Adverse Social Consequences." *Proceedings of the Institution of Civil Engineers - Transport* 169 (6): 353–65. <https://doi.org/10.1680/jtran.15.00073>.
- Luo, Wei, and Fahui Wang. 2003. "Measures of Spatial Accessibility to Health Care in a GIS Environment: Synthesis and a Case Study in the Chicago Region." *Environment and Planning B: Planning and Design* 30 (6): 865–84. <https://doi.org/10.1068/b29120>.
- Miller, Eric J. 2018. "Accessibility: Measurement and Application in Transportation Planning." *Transport Reviews* 38 (5): 551–55. <https://doi.org/10.1080/01441647.2018.1492778>.
- Neutens, Tijs, Matthias Delafontaine, Darren M. Scott, and Philippe De Maeyer. 2012. "An Analysis of Day-to-Day Variations in Individual Space–time Accessibility." *Journal of Transport Geography*, Special Issue on Time Geography, 23 (July): 81–91. <https://doi.org/10.1016/j.jtrangeo.2012.04.001>.
- Neutens, Tijs, Tim Schwanen, Frank Witlox, and Philippe De Maeyer. 2010. "Equity of Urban Service Delivery: A Comparison of Different Accessibility Measures." *Environment and Planning A: Economy and Space* 42 (7): 1613–35. <https://doi.org/10.1068/a4230>.
- Paez, Antonio, Christopher D. Higgins, and Salvatore F. Vivona. 2019. "Demand and Level of Service Inflation in Floating Catchment Area (FCA) Methods." Edited by Tayyab Ikram Shah. *PLOS ONE* 14 (6): e0218773. <https://doi.org/10.1371/journal.pone.0218773>.
- Páez, Antonio, Darren M. Scott, and Catherine Morency. 2012. "Measuring Accessibility: Positive and Normative Implementations of Various Accessibility Indicators." *Journal of Transport Geography* 25 (November): 141–53. <https://doi.org/10.1016/j.jtrangeo.2012.03.016>.
- Papa, Enrica, Cecilia Silva, Marco Te Brömmelstroet, and Angela Hull. 2015. "Accessibility Instruments for Planning Practice: A Review of European Experiences." *Journal of Transport and Land Use*, June. <https://doi.org/10.5198/jtlu.2015.585>.
- Pereira, Rafael H. M., Marcus Saraiva, Daniel Herszenhut, Carlos Kaue Vieira Braga, and Matthew Wigginton Conway. 2021. "R5r: Rapid Realistic Routing on Multimodal Transport Networks with R⁵ in R." *Transport Findings*, March, 21262. <https://doi.org/10.32866/001c.21262>.
- Pereira, Rafael H. M., Tim Schwanen, and David Banister. 2017. "Distributive Justice and Equity in Transportation." *Transport Reviews* 37 (2): 170–91. <https://doi.org/10.1080/01441647.2016.1257660>.

Venter, Christoffel. 2016. "Assessing the Potential of Bus Rapid Transit-Led Network Restructuring for Enhancing Affordable Access to Employment – The Case of Johannesburg's Corridors of Freedom." *Research in Transportation Economics* 59 (November): 441–49. <https://doi.org/10.1016/j.retrec.2016.05.006>.